I'm looking for a MQL4 programmer for long-term cooperation. The first Test Job. This document describes the TEST JOB.

FIFO is to be adhered, so trading server does not kick out orders for FIFO error. For the planned Cluster to evolve, we embed hooks/pipes/common file paths - whatever is available/ preferred so that multiple accounts may interoperate through shared data/information.

It has three parts:

1. The first part is to code a few trading related function calls.
2. The second part is to write a Block of Codes for saving trades related data sets on to files for subsequent optimization and interoperability between accounts (owned by same owner and running under FIFO).
3. The third part is to write only a structural block of a **Foundation EA** with the standard Nine(9) Code Blocks for External Inputs; Declarations of Internal Variables; Init; DeInit; Stop/Exit EA; a Housekeeping Block (saving in disk files, the trading history, alerts, errors, logs etc. for future use); a Block to Look at Chosen/ Required variables at N-ticks; a Block to Decide on Trading Actions; a Block for the Trading Actions; a Do No Trading Action Block (here wait for next tick or finish any unfinished housekeeping tasks). Additionally, each code bock will have a timer to gather execution time for that block and it will be printed while debug or may be turned off. The EA will have a Symbol or Instrument Rack, which means it will launch or work on N (2 to 7 symbols as inputted) symbols when launched. EA will launch from a dummy/proxy chart (other than any symbol on the Instrument Rack); close all other charts making sure there are no orders/positions in those symbols; and reopen the instrument rack’s symbol charts on M1 and attach the EA to all of them. EA will work on multiple time frame candles data internally (as inputted into EA - up to 5 – M1, M5, M15, M30, H1) irrespective of chart candle period being manually changed on the MT4 platform for looking at the screen. EA will open a dashboard/display window we call EA Information Table showing the Instrument Rack Summary Information Data on the launching dummy/proxy chart (illustration given below). EA will open on each active symbol chart a dashboard/display window (for that symbol) what we call Symbol Local Trades Table (illustration given below). The EA runs simultaneously on all the symbols using same logic but with different (symbol specific) input parameters and / or internal calculated variables.

Please put in copious amounts of commented (//) lines as notes and comments.

The above codes will be accepted by visual review and error free compilation and ability to just launch and navigate/operate the Foundation EA properly on a FIFO DEMO account on an MT4 platform without doing any real trading actions but showing that the EA is running and freewheeling and can do the housekeeping and there is stable flow through the various code blocks.

Budget, acceptance, funds release and project time/pace.

* I am Septuagenarian; decades ago, have programmed in few languages and did so very efficiently. Now, I am rusted with writer’s block, but I can read, understand, review and debug mq4 codes visually, logically and can provide constructive and useful feedback through my collaborative contribution into the project. Receiving ‘in-progress-mq4 files’ as often as MyFCC (my freelance cooperation coder) sends me but not to exceed a gap more than a calendar week is a requirement. Being Septuagenarian, I am slow and cautious. I must try not to propagate gaps / errors / blind spots / lack of understanding. Do little by little but do it correct. I hope to get this cluster done on a small budget by bootstrapping the 1st EA to do the 2nd EA so on so forth.
* My total long-term budget is 30 x $108 = $3240; of which $216 is allocated to this First TEST as initial commitment to get to know MyFCC personally, and then allocate 27 x $108 per EA = $2196 to the end and then a final closure bonus of $108.
* After First TEST, I shall commit $108 per EA and one EA at a time. Every subsequent EA will appear simpler and be completed easier or faster with more fun and more aha moments and joy, if we do the First TEST - FOUNDATION EA as the proper foundation.
* I keep my MQL4/5 community account at a level of $216 to release the payment per EA amounts on final acceptance of each EA. Progress payments, if any, they shall be no more than $54 each and must be linked to some progress marker.
* I like quality work; being patient, expecting steady progress in reasonable time windows. Example: 36 days for the First TEST-Function Code Blocks with FOUNDATION EA; then on the Cluster 1st EA (to be named ARENA) 27 days, 2nd EA (to be named BLINDMAN) 18 days, 3rd EA (to be named CARINA) 24 days so on so forth reaching a limiting turnaround time of say 9 days per EA towards the end of the cluster.

If my description is not clear, or you have question or query, please ask and I shall try to expand and explain with examples or illustrations.

**The First Part.**

**FEW TRADES FUNCTIONS**

1. MOOPPrice (string symbol, int type (for buy or sell), int N) – to return a double in broker digits.   
   (MOOPPrice stands for MyOldestOpenPositionPrice and N determines FIFO position N=0 oldest, N=1 is 1 after etc.)
2. MOOPSize (string symbol, int type (for buy or sell), int N) – to return a double in broker lot size (0.01 to 1.00)   
   (MOOPSize stands for MyOldestOpenPositionSize and N determines FIFO position N=0 oldest, N=1 is 1 after etc.)
3. MOOPProfit (string symbol, int type (for buy or sell), int N) – to return a double in broker digits.   
   (MOOPProfit stands for MyOldestOpenPositionProfit and N determines FIFO position N=0 oldest, N=1 is 1 after etc.)
4. MOOPSProfit (string symbol, int type (for buy or sell), int N) – to return a double in broker digits.   
   (MOOPSProfit stands for MyOldestOpenPositionSUM Profit and N determines FIFO position N=0 oldest position profit only, N=1 is sum of profits 2 positions oldest and 1 after, N=2 is sum of profits of 3 positions oldest and 2 after etc.)
5. MOOPSSize (string symbol, int type (for buy or sell), int N) – to return a double in broker digits.   
   (MOOPSSize stands for MyOldestOpenPositionSUM Size and N determines FIFO position N=0 oldest position size only, N=1 is sum of sizes of 2 positions oldest and 1 after, N=2 is sum of sizes of 3 positions oldest and 2 after etc
6. MNOPAPrice (string symbol, int type (for buy or sell), int N) – to return a double in broker digits.   
   (MNOPAPrice = MyNetOpenPositionAveragePrice and N determines FIFO position N=0 oldest, N=1 is 1 after etc.)
7. MOOPTicket (string symbol, int type (for buy or sell), int N) – to return int ticket# for my oldest open position.   
   (MOOPTicket = MyOldestOpenPositionTicket and N determines FIFO position N=0 oldest, N=1 is 1 after etc.)
8. MOOPTime (string symbol, int type (for buy or sell), int N) – to return a datetime.   
   (MOOPTime = MyOldestOpenPositionTime and N determines FIFO position N=0 oldest, N=1 is 1 after etc.)
9. Function calls to manage/calculate/update two arrays name-abbreviated as Array Double SNPP [HMI,HMTTK] and SNPPP[HMI,MaxNOP] .Both two dimensional double arrays are to be updated/calculated on every tick. Details are described on para#6 on page #11 of this document.
10. Close\_MOOP (string symbol, int type (for buy or sell), int N) – to close my oldest open positions   
    (Close\_MOOP= Close MyOldestOpenPositions (Respet FIFO) and N determines FIFO position N=0 close oldest only, N=1 closes 2 positions - oldest and 1 after , N=2 closes 3 positions and 2 after etc. closes (N+1) oldest positions under FIFO)

**A Vote Function:**

Please Create a function call such as :

int Vote (string symbol, int TF, int @otype, int @vote)

The above means that the function call will come in with the symbol name and TF of the candle and the reference of two variables - @otype as operation type BUY (OP\_BUY=0) or Sell (OP\_SELL=1) and @vote as the number of net votes in favor of the winning “otype” and returns 2 integers by reference or as a single integer like ( vote \*10 + otype) so we know the lowest position digit is either 0 or 1 and represents otype and we know the vote from the

* Look at last 5 candles of chosen TF (Default 5min) Candle 1 is just closed and Candle 5 is the 5th previous..
* In each Candle Conditions can be one of the following:

1. Green meaning Close is higher than Open.
2. Red meaning Close is lower than Open.
3. STOUT meaning Body is greater than both Head and Tail
4. Tall Head meaning Head is greater than both Body and Tail
5. Long Tail meaning Tail is greater than both Head and Body
6. Toji meaning Body is less than 27% of Height (High-Low) and Head and Tail are both greater than Body

Now look at voting logic and rules as explained using the table below.

* We reset/zero both Buy Vote and Sell Vote as we enter the function
* Each row represents a vote or a no-vote.
* Total possible number of unique votes are many but finite (likely 142,506/2). I have explained that after the table.
* However, we shall fill up 54 rows by our visually selected choices representing 108.
* We are filling the columns 1 to 5 and columns 7 to 12 with a selected candle condition (1 to 6) for the corresponding Candle Number (5 to 1). Columns 1-5 as a pattern decide a Buy Vote or no vote and columns 7-11 as a pattern decide a SELL vote or no vote.
* We can make a variable integer N\_Votes = 54 (or 9 or 18 or as we like) and can dynamically manage this table length
* For any row, if all 5 cells (1 to 5) are true then we add / increment count Buy Vote by 1, otherwise no change; similarly, if all 5 cells (7-11) are true then we add / increment count Sell Vote by 1, otherwise no change.
* We complete vote collection and counting till the N-Vote row.
* Here we have two choices – to return two integers – Buy Vote and Sell Vote or declared winner ‘otype’ and ‘vote’ as the winning difference the two votes.
* We can put in various checks such as – neither Buy Vote or Sell Vote can be greater than N-vote, and must be less than or equal to N-Vote

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Row # ↓ | BUY Votes: If for a row all 5 candle conditions match Buy gets 1 vote | | | | |  | SELL Votes: If for a row all 5 candle conditions match SELL gets 1 vote | | | | |
| Column # 🡪→ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|  | Candle 5 | Candle 4 | Candle 3 | Candle2 | Candle1 |  | Candle1 | Candle2 | Candle3 | Candle4 | Candle5 |
| 1 | 2 | 2 | 2 | 2 | 2 | M I R R O R  L I N E | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 1 |
| 3 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 |
| 4 |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |  |  |  |  |
| 13 |  |  |  |  |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |  |  |  |  |
| 16 |  |  |  |  |  |  |  |  |  |  |
| 17 |  |  |  |  |  |  |  |  |  |  |
| 18 |  |  |  |  |  |  |  |  |  |  |
| 19 |  |  |  |  |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |  |  |  |  |
| 21 |  |  |  |  |  |  |  |  |  |  |
| 22 |  |  |  |  |  |  |  |  |  |  |
| 23 |  |  |  |  |  |  |  |  |  |  |
| 24 |  |  |  |  |  |  |  |  |  |  |
| 25 |  |  |  |  |  |  |  |  |  |  |
| 26 |  |  |  |  |  |  |  |  |  |  |
| 27 |  |  |  |  |  |  |  |  |  |  |
| 28 |  |  |  |  |  |  |  |  |  |  |
| 29 |  |  |  |  |  |  |  |  |  |  |
| 30 |  |  |  |  |  |  |  |  |  |  |
| 31 |  |  |  |  |  |  |  |  |  |  |
| 32 |  |  |  |  |  |  |  |  |  |  |
| 33 |  |  |  |  |  |  |  |  |  |  |
| 34 |  |  |  |  |  |  |  |  |  |  |
| 35 |  |  |  |  |  |  |  |  |  |  |
| 36 to 49 Rows |  |  |  |  |  |  |  |  |  |  |
| 50 |  |  |  |  |  |  |  |  |  |  |  |
| 51 |  |  |  |  |  |  |  |  |  |  |
| 52 |  |  |  |  |  |  |  |  |  |  |
| 53 |  |  |  |  |  |  |  |  |  |  |
| 54 |  |  |  |  |  |  |  |  |  |  |
| N-Vote =54 |  |  |  |  |  |  |  |  |  |  |
| F[N,M] = 142,506 |  |  |  |  |  |  |  |  |  |  |

If any cell is empty or null – that cell condition is always true – so that unfilled rows cancel each other.

Further notes on this Vote Function:

Some Function F [M, N] is the total possible patterns of the following problem as I describe. I have M (=5) squares on a paper strip. I have unlimited supply of such strips. I have N (=6) cans of different colors. I can put only 1 color on any square and it can be any one of N (=6) colors. I must fill all M (=5) squares, and that will complete 1 strip and that pattern will be one pattern. If I keep doing this, a large (but finite) numbers of patterns will emerge; then there will be duplication. What is that number of unique / different patterns for this problem and that is my question! There is a closed form solution as an algebraic expression.

I am rusted and cannot remember the exact or correct solution. If I remember I shall let you know. My guess is (M\*N) CM = 142,506 but we shall manually choose and fill selectively 108 of them in 54 rows!

Our aim is to decide on our 1st entry position from our state SCS (Symbol Clean Slate) by using this Vote function as the EAs in the Cluster evolves one by one. However, in this TEST Job, in FOUNDATION\_EA we are using a fully described Vote logic with 3 candles (M=3) and 2 colors(N=2). Please see details on para(a) on page #25.2

Pease put in a program execution timer with a print line, so in this test we can get an idea of execution time for this function call.

**CREATE A TIME SERIES ARRAY.**

Name the Array to be Symbol\_History\_For\_DaDu\_Cluster or “S\_HFDDC” where S\_ is replaced by the symbol name.

It is two-dimensional array to hold double variables. Symbol\_History\_For\_DaDu\_Cluster [m,n] . It is big array.

The first dimension “m” takes care of various type of information and the “n” dimension holds the consecutive time values of the time series – we gather and create and save.

For our test let us make n = 362880 or enough to hold all the ticks for a full week. we shall test it running full week from Sunday 5pm to Friday 5pm

For our test let us make m = 27 or enough to hold the primary 4 variables (a) symbol\_ticktime(time of symbol\_tick), (b) Ask, (c) Bid and (d) Symbol\_Tick\_Volume plus ( m-4) calculated properties from the primary 4 variables.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | “n”--→ |  |  |  |  |  |  |  |  |
| “m” ↓ | Description ↓ | 0 | 1 | 2 | 3 | 4 |  |  | “n-1” |
|  | Values →→ |  |  |  |  |  |  |  |  |
| 0 | Symbol\_ticktime |  |  |  |  |  |  |  |  |
| 1 | Ask Price |  |  |  |  |  |  |  |  |
| 2 | Bid Price |  |  |  |  |  |  |  |  |
| 3 | Symbol\_Tick\_Volume |  |  |  |  |  |  |  |  |
| 4 | Property\_A  Spread |  |  |  |  |  |  |  |  |
| 5 | Property\_B Spread Max |  |  |  |  |  |  |  |  |
| 6 | Property\_C SpreadMax\_time |  |  |  |  |  |  |  |  |
| 7 | Property\_D SpreadMin |  |  |  |  |  |  |  |  |
| 8 | Property\_E SpreadMim\_Time |  |  |  |  |  |  |  |  |
| 9 | Property\_F Ask Max |  |  |  |  |  |  |  |  |
| 10 | Property\_G Ask\_Max\_Time |  |  |  |  |  |  |  |  |
| 11 | Property\_H Ask Mn |  |  |  |  |  |  |  |  |
| 12 | Property\_I Ask Min Time |  |  |  |  |  |  |  |  |
| 13 | Property\_J Bid Max |  |  |  |  |  |  |  |  |
| 14 | Property\_K Bid Max Time |  |  |  |  |  |  |  |  |
| 15 | Propert\_L Bid Min |  |  |  |  |  |  |  |  |
| 16 | Propert\_M Bid Min Time |  |  |  |  |  |  |  |  |
| 17 | Property\_N Ask\_Rate |  |  |  |  |  |  |  |  |
| 18 | Property\_O Bid\_Rate |  |  |  |  |  |  |  |  |
| 19 | Property\_P  Ask\_Force |  |  |  |  |  |  |  |  |
| 20 | Property\_Q Bid\_Force |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| “(m-1)” | Property\_X |  |  |  |  |  |  |  |  |

We shall define few properties for our test, but the code should be able to handle “m-4” properties when we define them explicitly as an expression / calculation.

Property\_ A is Spread = Ask-Bid for that symbol\_tick\_time or same column.

Property\_B is SpreadMax is the maximum value of the Spread from the row for spread.

Property\_C is Spread\_Max\_Time is fetched from the 1st or “0th” row of times, when it occurred most recently.

Property\_D is SpreadMin is the minimum value of the Spread from the row for spread.

Property\_E is Spread\_Min\_Time is fetched from the 1st or “0th” row of times, when it occurred most recently.

Property\_F is Ask Max is the maximum value of the Ask from the row for Ask.

Property\_G is Ask\_Max\_Time is fetched from the 1st or “0th” row of times, when it occurred most recently.

Property\_H is Ask Min is the minimum value of the Ask from the row for Ask.

Property\_I is Ask \_Min\_Time is fetched from the 1st or “0th” row of times, when it occurred most recently.

Property J K L M are same as F-G-H-I as applied on Bid instead of Ask.

Property N is Ask\_Rate. It is calculated as (Ask at current tick - Ask at previous tick) (It will be great to color the cell green for positive and red for negative and yellow for zero)

Property O is Bid\_Rate. It is calculated as (Bid at current tick - Bid at previous tick) (It will be great to color the cell green for positive and red for negative and yellow for zero)

Property P is Ask\_Force. It is calculated as (Ask\_Rate at current tick – Ask\_Rate at previous tick) (It will be great to color the cell green for positive and red for negative and yellow for zero)

Property Q is Bid\_Force. It is calculated as (Bid\_Rate at current tick – Bid\_Rate at previous tick) (It will be great to color the cell green for positive and red for negative and yellow for zero)

Few Sanity Check and storage management awareness are required:

Chosen default dimension 27 x 362880 x 4 bytes/cell would take up memory storage of near about 40MB per “S\_HFDDC” symbol array. If an account or MT Platform uses 5 symbols and if a computer or vps runs 3 platforms we are taking away about 600 MB of memory from the device by allocation for this Array.

[Q1 is 4 bytes/cell a correct assumption?]

This array keeps information from more than a week at symbol tick intervals. It would be adequate to keep the latest 27 hours in memory and everything prior to go in disc files in the common\_data\_path which is a call to TerminalInfoString(TERMINAL\_COMMONDATA\_PATH)

(example C:\xxxxxx\yyyyy\zzzzz\AppData\Roaming\MetaQuotes\Terminal\Common etc etc. ) and we make a discipline or practice to save this array in properly named disc-files in that common data path in weekly files starting 17:01 hours EST Sunday and ending on 17:01 hours EST Friday. However, for in-current-week reboot / restart etc. we keep writing to the current week’s disc file every 5 minutes or 288 times daily we can recover with only small gaps.

The file path and name is 🡪Common\_data\_path\HFDDC\ “Symbol”+DDMMYYYY for the Sunday when the week starts.

Another sanity check point, when it is run for the very first time – the array is all null / zeroes. Is there a way to seed the top four rows from somethings available from MQL or the Trading server as – predefined and already preserved?

OnTick we are working on / filling only the 1st or “0th” column. But just before that, we must save / keep everything/every cell and column by shifting to the next higher column - the right most column vanishes- but in common data path disc file we shall have everything captured, if the routine is running.

**The Second Part.**

Please write a code block for saving trades related data sets on to files.

**File Structure of Saved / Gathered Information (Five groups of Information!)**

1. Files Structure of saved/ gathered information of our EAs will be same/ similar / common for this EA Cluster.
   1. The five types of data/information –Orders, States, Errors, Reports, and Performance will be fine and enough for this cluster and future optimization tuning. Example: For HMI ( How Many Instruments in Instrument Rack) =4 there will be 20 folders. Each folder will have 1 file weekly.
   2. We choose our own Cluster Data Folder path as “Cluster\_Data\_Path “ and choose filename convention as shown below:   
      Cluster\_Data\_Path \Account#\EANAME\Symbol\Reports\ filename.txt/csv, where filename is same ( DDMMYYYY for the Sunday which starts the Week) for that whole week for all groups/types of data. Example: For the week of June 23 -28, 2019, all filename = 23062019.
   3. In the Third Part you will see some examples of rows and columns of each type of data files as a part of description / specifications of the FOUNDATION EA over next few pages.

**The Third Part.**

Description and Specifications of the FOUNDATION EA. This has two Sections: (a) Structure and (b) Embodiment Example

SECTION (a) STRUCTURE.

General Overall Description: Each EA will be mq4 source program, that will run on an MT4 Platform. At launch, it will have a set of inputs. While running, it will have a set of Reports / Logs / Alarms / Alerts as outputs. While running, it will perform the required trading actions of opening, tracking and closing various trades (positions) in specified (by input) symbols/instruments according to the structure and logic provided in this “Strategy Document” (a) respecting Government Rules and Regulations such as FIFO for USA based account owners, (b) smoothly interfacing and interacting with the Trading Servers to avoid too many errors and rejections and retries for strategy order placements and their executions, (c) enabling smooth auto-restart in case power and internet outages, and (d) ability to save / store in files the performance data /results over preselected (by input ) time frames to perform periodic adaptive tweaks (optimization) on the input parameters, by treating them as “internal to strategy” variables using additional optimization (tweak) logic to be provided subsequently as the cluster moves to B, C, D ….Z, AA. .

**The Structure**: The Structure is very important. Structure will be same or mostly same for all EAs in this cluster. Beginning with this FOUNDATION EA , we shall focus and pay attention to our naming and comments conventions. The structural blocks will be (1) External Input Management,; (2) Internal Variables Definitions and Management, especially initializations and resets ( init, deinit, restart etc); (3) Symbols Chart Displays and Management, (4) Saving of required Data/Information in disk files for subsequent uses; (5) Management of Alerts, Errors, Exceptions for early manual interventions and actions required from outside of EA such as debug, update, relaunch, restart so on so forth ( these 5 blocks, we may collectively call housekeeping), the following ones are the work or actions of the EA in the major OnTick Block), (6) Look (gather information to anchor where we are); (7) Calculate, Decide, Store, Plan Forward; (8) Trading Actions ( enter, close, modify orders and positions); (9) Catch up housekeeping, Do nothing and wait for reentering block(6). We have described / planned / chosen the structure to have above nine (9) blocks of codes for our communication and coordination for reuse and consistency as the cluster progresses.   
Goal is to stay within this chosen structure for the full cluster.

**Naming of Variables:** We shall name our external and internal variables as same and common for the cluster. We shall use descriptive common English like underscored names. For long variable names, when we abbreviate, we shall write down in comments – what it means on their first occurrences and keep it same for the cluster.

Declarations of variables internal and external and notes on their initialization, resets and global referencing shall be coded in two distinct blocks and will have much and clear comments.

**Below are the common External (Input parameters) and Internal (local and used) variables for the Cluster: (**few may be added or removed from individual EAs**)**

External Inputs.

1. Intger MyMagicNumber; MyPA; MyLTI; MyDTI, MyATI (MyMagicNumber is identifier, MyPA is Paired Account #, MyLTI /MyDTI/MyATI are tick interval control counters)
2. Integer HMS 🡪 Default = 7 (HMS stands for How Many States)
3. Integer MaxNOP, SPANOP, SIRNOP 🡪 maximum number of open trades(positions) in a symbol in different States, Default =5 (General SCS Cycle Start), 15 (for states holding Profit ), 30( for states holding Loss)at start but as the account balance/equity grow – they are reset/computed/updated, when in “Symbol\_Clean\_Slate” State.
4. Integer MyTradeSize 🡪 Default 1 (meaning 1k) at start but as the account balance/equity grow – it is computed/updated, when in “Symbol\_Clean\_Slate” State.
5. Integer array TSM [MaxNOP] 🡪 Default at Start [1, 1, 2, 3, 5 ,5, 5, …….5].   
   TSM stands for “Trade Size Multiplier, used for multiplying “MyTradeSize” to obtain specific TradeSize
6. Integer HMI 🡪 (How Many Instruments) – Default 4, it is used to fill the “Instrument Rack” Details will be described below:
7. Integer HMTTK 🡪 (How Many Ticks to keep the Values of NPP[]) Net Position Profit for each symbol.
8. Integer MTCS 🡪 Minutes to Clean Slate – Default 7182 – Weekly New Horizon! Any trade shall auto close, if not already closed after MTCS minutes after opening.
9. datetime TTOT 🡪 Time to open auto trading – default USA Time Sunday 5:10 PM; datetime TTCT 🡪 Time to close auto trading – default USA Time Friday 4:50 PM No auto-trade-opening/market entry after TTCT before TTOT
10. Integer TF1, TF2, TF3, TF4, TF5 🡪 Time Frame TF1 in minutes 🡪 Default 1; Time Frame TF2 in minutes 🡪 Default 5; Time Frame TF3 in minutes 🡪 Default 15; Time Frame TF4 in minutes 🡪 Default 60; Time Frame TF5 in minutes 🡪 Default 15; [ But we can use these 5 time frames in different ways and contexts in different EAs in the cluster)
11. String Cluster\_Data\_Path 🡪 “ C:\DaDu\_Gift\´ 🡪 to direct EA Data files to this path or MQL Default =TerminalInfoString(TERMINAL\_COMMONDATA\_PATH)
12. Boolean GREED, Default=false; double RecovPoint, Default = 100;
13. More Inputs may be added in future. Strategy Set (Plan 27) Set A……………………to………………Set Z & Set\_AA

Internal local variables defined, used, saved and reported/journaled by the Strategy:

1. Boolean MyHedging 🡪 default = False (as US is FIFO), as initialized – but keep for future.
2. Array double Delta1[ MaxNOP] 🡪 Initialize at Start [ 0.0001, 0.0001, 0.0002, 0.0003, 0.0005,..]
3. Create / Define as any many as HMS (example 7 in EA\_A given below) 1 dimensional Boolean arrays sized to HMI ( as many as symbols)   
   or create / define a Boolean Array States [i,j] where1st dimension *i* holds/represents the “string names of the states” and 2nd dimension j holds/represents the string names of the Symbols. They have to updated every tick, reset/initialized at SCS state. Example of EA\_A is given below.  
   1. Boolean SCS (Symbol\_Clean\_Slate) = default = True, as initialized – checked every tick.
   2. Boolean SWA (Symbol\_Walk\_Ahead) = default = False, as initialized – checked every tick.
   3. Boolean SPTL (Symbol\_Pause\_To\_Look) = default = False, as initialized – checked every tick.
   4. Boolean SSA (Symbol\_Sprint\_Ahead) = default = False, as initialized – checked every tick.
   5. Boolean SRCOIP (Symbol\_Reversal\_Close\_Oldest\_in\_Profit) = default = False, as initialized – checked every tick.
   6. Boolean SHIL (Symbol\_Hold\_In\_Loss) = default = False, as initialized – checked every tick.
   7. Boolean SIR (Symbol\_in\_Recovery) = default = False, as initialized – checked every tick.
4. Integer NOP (Number Open Positions) 🡪 value can be 0 to MaxNOP – calculate this on every Tick
5. Create Display Tables 🡪 **Local Trades Tables** (one per Instrument Symbol) Default Size is (12 x (MaxNOP+6)).  
   Example EURUSD\_LTTable [ 12 Columns x (MaxNOP+6) rows)   
   The 12 columns are listed below:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Order #  By EA / Server | Order Place Time | Order Exec. Time | Order Place Price | Order Entry Price | Order Position Type | Order Position size | Index of Position | Symbol  Price Now | Order Position Cost | Order Position Profit | Remarks |
| xxxxxxxx |  |  |  |  |  |  | 0 |  |  |  | Oldest |
| xxxxxxxx |  |  |  |  |  |  | 1 |  |  |  |  |
| xxxxxxxx |  |  |  |  |  |  | 2 |  |  |  |  |
| xxxxxxxx |  |  |  |  |  |  | 3 |  |  |  |  |
| xxxxxxxx |  |  |  |  |  |  | 4 |  |  |  | Latest |
| Blank | Blank | Blank | Blank | Blank | Blank | Blank | Blank | Blank | Blank | Blank | Blank Row |
| MaxNOP | 5 |  |  |  |  |  |  |  |  |  |  |
| Blank | Row |  |  |  |  |  |  |  |  |  |  |
| Sum-All | NOP |  |  | Avg Price |  | Total |  |  |  | Total  NetPL | ∑  0 to NOP-1 |
| Sum-all but Last | NOP-1 |  |  | Avg. Price |  | Total  -Last |  |  |  | Sum  NetPL | ∑  0 to (NOP-2) |

Display this table on a fresh Tab/Sub-Window/Area which is like the “Terminal🡪 Trades Tab Window for Symbol.

1. Array Double SNPP [HMI,HMTTK] and SNPPP[HMI,MaxNOP] are both two dimensional double arrays to be updated/calculated on every tick.   
   SNPP is Symbol Net Position Profit and SNPPP is Symbol Net Position Progressive Profit.  
   SNPP is Symbol Net Position Profit and SNPPP Symbol Net Position Progressive Profit. Both are 2 dimensional.

First dimension size is fixed at 9 and we initialize to fill with HMI (=4) and then do not need resize them while running the tick loop in EA, gets initialized at init and deinit loops.   
Second dimension in both SNPP and SNPPP are zeroed at init and at SCS ( of the corresponding symbol in the 1st index)  
Second dimension of SNPP is always HMTTK ( default=1024), on a tick, index 1022 to 0 is shifted to index 1023 to 1 and then the current at-tick value of the sum of profits of all open positions in this symbol is saved in SNPP[\_symbol,0] and oldest value SNPP[\_symbol,1023] is lost.   
Second dimension of SNPPP is only filled up to ‘NOP = number of positions opened in the symbol in the running cycle until the next SCS’, which is less < 99 and <= SIRNOP (default 30). On tick, all the NOP cells from SNPPP[symbol, 0 to (NOP-1)] are recalculated and saved. The index (nop-1) hold the sum of profits of opens positions   
The differences / similarities/ common elements between SNPP and SNPPP are as follows:

* 1. SNPP [\_symbol, 0] always holds the sum of profits of all open positions of the symbol which is equal to SNPPP[\_symbol, (nop-1)]
  2. SNPPP[\_symbol, n] where n goes 0 to (nop-1) are calculated/ updated ‘nop’ times on every tick and fills SNPPP[\_symbol, n] = sum of latest (n+1) positions.

SNPPP[\_symbol,0] holds the profit of the latest( most recent position); SNPPP[\_symbol,1] holds the sum of profits of the latest 2 positions, etc. etc. and SNPPP[\_symbol,(nop-1) holds the sum of profits of all open positions, SNPPP[\_symbol, (nop-2)] holds the sum of profits of all open positions but the oldest one…so on so forth

SNPP array is initialized/zeroed at EA “init” and when the EA comes to the state we call “ SCS (Symbol\_Clean\_Slate) “; then with open position(s), this array keeps the history over last HMTTK ( How Many Ticks to Keep=1024) ticks of the Symbol\_Net\_Position\_Profit like a shift register or push-down stack – index(0) is current tick, Index (1) is last tick – so on so forth These are calculated / updated on every tick

SNPPP array is initialized/zeroed at EA “init” and when the EA comes to the state we call “ SCS (Symbol\_Clean\_Slate) “; then with open position(s), this array keeps the progressive history over open positions. Symbol\_Net\_Position\_Progressive\_Profit is like a shift register or push-down stack with respect to age of the positions in the net total position – index (0) is sum of all open positions, index (1) is all older positions but the last/latest one, index (2) is all older positions but the last 2 positions – so on so forth. These are calculated and updated on every tick.

Array double Symbol\_OrderCost [MaxNOP] 🡪 This array is initialized/zeroed at EA “init” and when the EA comes to the state we call “SCS (Symbol\_Clean\_Slate) “; then with open position(s), this array keeps the history of currently open orders/positions – this array is the same column # 10 of the Table – we call Local Trades Table - after header, next MaxNop Rows.

1. datetime 🡪 CycleStartTIme, CycleEndTime
2. double 🡪 CycleTime, CycleProfit, CycleCost, AverageCycleTime, AverageCycleProfit, Average CycleCost. WeeeklyProfit, DayEndBalance, DayEndEquity. DayEndProfit.
3. Integer 🡪 RTTradesThisCycle, RTTradesThisWeek
4. Boolean JCC= Just Closed Candle 🡪 is triggered every new candle of TF1, TF2,TF3 to capture “performance” data to be saved in files.
5. Double SPRS5Y = Symbol Profit Realized Since 5pm Yesterday
6. Double SPRSWS = Symbol Profit Realized Since Week Start.

**States** of the Symbol/Instrument in the Account under EA are a common, crucial and pivotal boolean entities. We name them and use them in consistent / analogous / similar ways within this cluster from EA-A to the next EA-B. Our defined States hold the structure of the EA.

We call 1st State SCS or Symbol Clean Slate when, in the account, there is no open position or no open order and margin used by the Symbol is zero, then the State SCS is true(=1) otherwise false(=0). This is an anchor and common state with same name and same meaning in all EAs of this cluster. The objective is to arrive at this state as often as we can with desirable net profit in that symbol or a desired exit with an acceptable net loss. The EA exits this SCS state as soon as an entry position is taken in that symbols and travels through other states (defined in each EA) and strives to reach SCS at an opportune moment.

We enforce a structural discipline with respect to how many states we define or declare. With open position (s) we may be in profit or in loss, so we need at least two more states one in net profit and one in net loss. We shall have minimum 3 states, but we cap the maximum number of states to 9. We hope to keep this symmetrical or nearly symmetrical by using same number of states while in profit or in loss. For our own discipline we define an external integer input NOS (Number of States), and let that be 3,5,7,9 etc.

Example: EA\_ARENA has seven (7) states and EA\_BlindMan has five (5) states. This means that for an EA, various actions are described in terms of those defined states and conditions. To emphasize, each EA wants to leave the state SCS when in SCS by entering a position which marks the start of a trading cycle, and when in other states holding position(s), the EA wants to close and enter the SCS State, which marks the end of a trading cycle. Trading cycle data will be captured by EA.

TICK INTERVAL COUNTER USE.

In addition to using MyMagicNumber and MyPA for pairing an account for interoperability, we plan for use of tick counters within the OnTick loop for different kind of “actions and decisions” we need to take for the EAs. Certain actions will always be on every tick, some actions we call as “to look at” those will be done at every MyLTI ticks, some actions we call as “ to decide” – those will be done at every MyDTI ticks and trading actions will be done at MyATI ticks.

MyLTI <= MyDTI<=MyATI and for the EA\_ARENA, our default values are MyLTI =MyDTI=MYATI =1.

**The Instrument Rack**: An EA is usually attached to a symbol on the MT4 Station chart of that symbol; but we envision this EA to use minimum 2 symbols and maximum of 7 symbols. The idea being it will be more than solo music but like an orchestra/symphony. For our own discipline we define an external integer input HMI (How many Instruments), and let that be 2,3,4,5,6, or 7. We pre enumerate the seven instruments /symbols as EURUSD, GBPUSD, AUDUSD, EURCAD, EURGBP, USDCAD, GBPCAD. We also pre assign alternate Trade Type (Buy or Sell) for the above symbol for an EA to start with Buy, Sell, Buy, Sell, Buy, Sell etc.. However, it will be clear that our EA logic will be able to change them as an internal variable due to interoperation and adaptation while in the state SCS.

Example: if HMI = 4 then the first 4 symbols (EURUSD, GBPUSD, AUDUSD, EURCAD) are in our EA will start as.

|  |  |
| --- | --- |
| Instrument Name | Trade\_Type |
| ERUUSD | Buy |
| GBPUSD | Sell |
| AUDUSD | Buy |
| EURCAD | Sell |

On that launching proxy window – EA will display the Intrument\_Rack with few other variables as they are on real-time.   
Example :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instrument Name | Trade  Type | State | Net Pos Size | Net Price | Net P/L | Realized Weekly P/L |
| EURUSD | Buy |  |  |  |  |  |
| GBPUSD | Sell |  |  |  |  |  |
| AUDUSD | Buy |  |  |  |  |  |
| EURCAD | Sell |  |  |  |  |  |

**File Structure of Saved / Gathered Information (Five groups of Information!)**

1. Files Structure of saved/ gathered information of our EAs will be same/ similar / common for this EA Cluster.
   1. The five types of data/information –Orders, States, Errors, Reports, and Performance will be fine and enough for this cluster and future optimization tuning. Example: For HMI =4 there will be 20 folders. Each folder will have 1 file weekly.
   2. We choose our own Cluster Data Folder path as “Cluster\_Data\_Path “ and choose filename convention as shown below:   
      Cluster\_Data\_Path \Account#\EANAME\Symbol\Reports\ filename.txt/csv, where filename is same ( DDMMYYYY for the Sunday which starts the Week) for that whole week for all groups/types of data. Example: For the week of June 23 -28, 2019, all filename = 23062019.
2. Below let us make some examples of rows and columns of each type of data files starting with “Orders”
   1. Our saved “Orders” File is a coma delimited flat file with a header row as below – it is basically a weeklong history of our Trades with the closing information as every week will start and end with SCS. (as a choice / requirement)   
       I am not sure – if we can file write all 12 columns in one row or we must write them in two rows one for the opening time/ side and one for the closing time/side and then sort and merge them into one row.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Col-1 | Col 2 | Col 3 | Col 4 | Col 5 | Col 6 | Col 7 | Col 8 | Col 9 | Col 10 | Col 11 | Col 12 |
| Order #  Ticket# | Order Place Time | Order Exec. Time | Order Place Price | Order Entry Price | Order Position Type | Order Position size | Order Close Price | Order Closed Price | Order Position Cost | Order Position Profit | Order Close Time |
| Same # | Fill | Fill | Fill | Fill | Same | Same |  |  | Same |  |  |
| Same # |  |  |  |  | Same | Same | Fill | Fill | Same | Fill | Fill |

* 1. Our saved “States” File is a coma delimited flat file with a header row as below – it is basically a weeklong history (a time lapse map) of “States Transitions” for a Symbol under the EA with time interval of TF1 or less. For a week, there will be 7200 rows or more. At an interval of TF1 a row will be filled, even if the ‘State’ is same or if the “State” Changes in between that interval, a row will be filled.
     1. The States column may be Strings like SCS etc. or enumerated them as 1,2,3,4 ,5,6,7
     2. I a row, if there are NOP positions, for that row there will be (NOP+6) columns – see below.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Col-1 | Col 3 | Col 3 | Col 4 | Col 5 | Col 6 | Col 7 | Col 8 | Col 9 | Col 10 | COLs  11 to (NOP+5) | | | | | | | Col (NOP +6) | |
| Date Time | Acct.  Equity | States | NOP | Net  Position  Size | Position Type | SNPP[0] =  SNPPP  [NOP-1} | SNPPP  {NOP-2] | SNPPP  [NOP-3] | SNPPP  [NOP-4] |  |  |  |  |  |  | SNPPP[0]  Profit of oldest position | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |

* 1. Our saved “Errors” file is a coma delimited flat file with a header row as below – it is a weeklong errors-alerts history (errors alerts) of a Symbol under the EA. A row is entered when an error or alert is encountered.

|  |  |  |
| --- | --- | --- |
| Date Time | Error Code | Remarks for error and alerts |
|  |  |  |
|  |  |  |

* 1. Our saved “Reports” file is a calculated/gathered reports file using other saved data files. It is a weeklong progress report of a Symbol under the EA. The Reports is created / opened at 4:59 pm NY Time on Sunday and closed/saved at 5:01 pm, filling the 1st row and last row respectively, and when we either enter the state SCS or a New Hour more than 1 hour after last SCS or 5 pm , we add a new row to this file. There will be more than 120 rows and likely less than 720 rows in a week.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Col-1 | Col 2 | Col 3 | Col 4 | Col 5 | Col 6 | Col 7 | Col 8 | Col 9 | Col 10 |
| Date Time | Equity | Balance | Cycle Number | This Cycle Time | This Cycle Profit | Avg Cycle Time | Avg Cycle Cost | Daily Profit | Weekly Profit |
| xxxx1659 | EQU | BAL | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| xxxx1701 | EQU | BAL | 0 |  |  |  |  |  |  |

* 1. Our saved “Performance” file is a weeklong running history of performance of a Symbol under the EA. We enter a row at start of TF1, TF2 and TF3 candles. It is a large file with (7200 +1440 + 480 =)9120 rows of data.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Col-1 | Col 2 | Col 3 | Col 4 | Col 5 | Col 6 | Col 7 | Col 8 | Col 9 | Col 10 | Col 11 | Col 12 |
| Date Time | Candle TF | JCC  Color | JCC  Open | JCC  Close | JCC Length | EA otype | Net Position Size | Net Position  Profit | SPRS5Y | SPRSWS | Acct  Equity |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

JCC= Just Closed Candle. SPRS5Y = Symbol Profit Realized Since 5pm Yesterday SPRSWS = Symbol Profit Realized Since Week Start.   
Please note the five saved files have lots of duplication and redundancy. They will help us in debug / test and optimization and implementing the EA cluster going forward.

More General Notes:

I summarize the cluster strategy theme; then as subsequent EA algorithms emerge, we code within the ‘sandbox’ or template of the FOUNDATION EA:

* Aim no greed and no fear –keep margin used low and free margin is high. Low Greed. Consistent steady small profits. No Hedging. Focus on and adhere to FIFO. Oldest position is to be closed first.
* Neither hold loss nor profit for too long and reach clean slate as often as practical.
* We are not using various available/stochastic/predictive indicators or signals.
* We shall not put a Stop Loss (SL Line) in Trading Server while in loss territory
  + We shall mange those by EA’s active FIFO Closing of open position(s)
  + Only when in Net Profit is above Minimum Profit, we put in trailing step stop.
* We keep sharp focus on just one current Cycle at a time.

The elements are: Calculations or Action or Housekeeping

* + Symbol Net Position Profit in Account Currency. (Calculation)
  + Symbol Net Position Size in 0.01 (1k unit) (Calculation)
  + Symbol Net Position Profit in distance (points) or in account currency (Calculation)
  + Symbol Any Specific (oldest, latest, oldest2, latest 2 etc) positions-set Net\_Sum\_Profit. (Calculation)
  + Close Symbol Oldest Position (s) by number of positions or by amount of size to be reduced. (Action)
  + Do Clean Slate. Close open positions and pending orders and reset required variables. (Action)
  + Open a Symbol Position at Market of desired type and size (Action)
  + Create a Symbol Pending Order of desired type, size and price (Action)
  + Gather and save Data/information in structured disc Files for learning, review, debug and optimization (Housekeeping)
  + Handle and Manage Errors, Alerts, de\_inits, interoperability and restarts ( Housekeeping & Actions)

The above function calls / code blocks will form a core common set code in all EAs of the cluster.   
Hope in general simple way, I have communicated how we hope to be correct and successful. If any difficulty, please ask questions and I shall try to explain / express in another way. Please aim to finish Stragey\_FOUNDATION EA as above and as described in this Strategy document.

Please read and give me your feedback of your understanding and commonalities and ask questions, if anything is not clear. As we learn working together, you will see each subsequent Strategy write ups will be more rapidly and easily understood and coded by you!

Each subsequent Strategy will have more common elements from the previous ones in the series/cluster.

Please send me in-progress updates on this TEST JOB on mq4 files for me to look at, read, understand and provide feedback.

Please read the complete document all pages. Duplication is there for redundancy and self-check and as potential comments to add in the actual code script.

Pease keep/maintain input selection of choices to log / print to Journal / print to file(s) / email /Text-SMS (with time stamp and remarks), the following: (a) Alerts, (b) Errors, (c) Reports, (d) Orders-placement, executions (open- close- modify-retries-failures).

These will help the debug process at the beginning and the help optimize when move along with the EAs one by one!

1. Alerts give an audible alert ( or may be tuned off) and sends ( or may be turned off) an email to [sukumar@sikdar.com](mailto:sukumar@sikdar.com) and sends ( or may be turned off) text SMS to +1 484-888-6045 and always prints to an Alerts file.
2. The file paths and conventions are already described.
3. When we enter SCS (Symbol Clean Slate) State, we capture and write to file data/results related to items and variables to captured and saved.
4. arrows and connecting lines are created for proper visualization. Up Green arrow for Buy Entry, Down Red arrow for Sell Entry, Open and Close of a completed RT trade are connected by a dotted line and a profit transaction gets green color and loss transaction gets a red color.
5. Show the net position (solid Green) lines and TP (dotted Green) lines on the respective symbol chats
6. This EA runs on an MT4 platform logged into one account and conducts the same tune (logic) on few (4 in our Example case EA\_A=ARENA) instruments.
7. This EA itself launches the respective HMI symbol/instrument windows, if not already open. It can be launched / initiated from any (proxy / dummy instrument) window on that platform. EA does use it own set of (4 or so) symbols/instruments and does not get effected by what chart time frame is currently open, as it uses own input of time frames (TF1, TF3, TF3, TF4 etc. as input parameters).
8. We shall open an unused symbol window and launch the EA from there and it would then open/launch the other chart windows as tabs for the instruments on our Instrument Rack. With the EA running, there will be only HMI + 1 (the launching/dummy window) chart windows open and any other open non used chart windows will be closed.
9. On that launching proxy window – we display the Instrument Rack and a few other variables as they are on real-time.   
   Example:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instrument Name | Trade  Type | State | Net Pos Size | Net Price | Net P/L | Realized Weekly P/L |
| EURUSD | Buy |  |  |  |  |  |
| GBPUSD | Sell |  |  |  |  |  |
| AUDUSD | Buy |  |  |  |  |  |
| EURCAD | Sell |  |  |  |  |  |

1. We create few (4 as shown above) additional tabs for individual windows to display our Local Trades Tables as EURUSD\_LTTable and similar GBPUSD\_LTTable , AUDUSD\_LTTable, and EURCAD\_LTTable.
2. Please create two function calls to calculate some profit values (on current tick)
3. Call it Symbol Specific Position Profit or SSPP (string symbol, integer num) where symbol takes the symbol name and num takes the position number, and we know position number 0 is the oldest position and position number (nop-1) is the latest so on so forth and the function returns the value in account currency with 2 decimal accuracy.
4. Call it Symbol Specific Position Cluster Profit or SSCPP (string symbol, integer num1, integer num2) where symbol takes the symbol name, num1 takes the position number to begin cluster and num2 takes the position number to end the cluster and we know position number 0 is the oldest position and position number (nop-1) is the latest so on so forth and the function returns the value of the sum of the profits held by position num1 to num2 in account currency with 2 decimal accuracy
5. For our evolving cluster ( A to Z and AA) of 27, please start thinking of local inter-operation and coordination of information between two or more MT4 platforms each connecting to trading servers ( same or different) with different account numbers (one ownership) as all the platforms are running on same machine ( desktop or VPS) able to connect with same storage ( C Drive or Defined Drive folders)
   1. I feel I should explain “interoperability” as I see it, so you and I see the same goal, path and actions. Here is my description:
   2. Our EA Cluster must obey FIFO, as US based accounts with No Hedging and no holding or ordering opposite type positions in same symbol at the same time. ‘Interoperability’ is our way of achieving pseudo hedging by running same EA on two Accounts as either can flip or not flip the Buy/Sell Type on the symbol to the opposite, when in SCS State, if account is paired and if the other account is running the EA by checking the Symbol States on the other Account.
   3. The interoperation flipping logic is in the EA and is EA specific.
   4. Structurally, it would be a code bock same in structure on all EAs of our cluster.
   5. Example: In our FOUNDATION\_EA: (FOUNDATION has seven symbol states)
   6. The deciding account, when in Symbol State SCS checks and does the following:
   7. If paired account symbol state is also SCS, neither account flips symbol Buy/Sell Type, keeps same, no changes.
   8. If paired account symbol state is SPA, deciding account changes its symbol Buy/Sell Type to that of the other.
   9. If paired account symbol state is SIR, deciding account changes its symbol Buy/Sell Type to opposite of the other.
   10. If paired account symbol state is SWA, deciding account changes its symbol Buy/Sell Type to opposite of the other.
   11. If paired account symbol state is SRCOIP, deciding account changes its symbol Buy/Sell Type to opposite of the other.
   12. If the paired account symbol state is SIR, deciding account changes its symbol Buy/Sell Type to opposite of the other.
   13. If the paired account symbol state is SHIL, deciding account changes its symbol Buy/Sell Type to opposite of the other.
6. “Interoperability” is between two (but can be more) accounts, by same owner, running the same EA, on different MT4 Platforms, all running on same desktop or server (common C: drive). Accounts are paired. Each account knows its number and its pair’s number. Both must be running the same EA for the pairing to remain in force. Both accounts run the same EA. Both accounts can read or check the latest /current status of the Symbol States of the other account. The paired account’s number will be an external input entered at launch time. String common\_data\_path=TerminalInfoString(TERMINAL\_COMMONDATA\_PATH)In the new MQL4, FILE\_SHARE\_WRITE and FILE\_SHARE\_READ flags should explicitly be specified for shared use when opening files. If the flags are absent, the file is opened in exclusive mode and cannot be opened by anyone else till it is closed by the user who opened it

Section (b): FOUNDATION EA : Embodiment Example.

Strategy Name: FOUNDATION\_EA

A few Definitions and Inputs:

Include all the common defined / declared external inputs and internal variables of “Section(a) of the Third Part” of this document.

The “Strategy mq4 File” is a software program; it will work and follow different paths / logic sequences. It is a closed, bound, finite and repeatable system/machine which will keep coming back to the well-defined “States” or “Set of Conditions and Boundaries”.

Below are our defined “States” for this FOUNDATION\_EA Strategy (please build and keep track of our convention of naming of inputs and local/global variable to eventually make 27 Strategies as a Cluster or a Set🡪 A to Z, AA)

1. “Symbol\_Clean\_Slate” is when there is no open position, symbol-used-margin is zero and margin available is equal to account equity. (SCS)
2. “Symbol\_Walk\_Ahead” is when there are open positions (of course they are of same type for the same instrument/symbol) and Symbol\_Net\_Position\_Progressive\_Profit(0) is higher than sum of “all but the last/latest” Symbol\_positions profits, which is Symbol\_Net\_position\_Progressive\_Profit (1). (SWA)
3. “Symbol\_Pause\_To\_Look ” is when there are open positions (of course they are of same type for the same instrument/symbol) and Symbol\_Net\_Position\_Profit (0) is lower than sum of “all but the last/latest” Symbol\_positions profits. (SPTL)
4. “Symbol\_Sprint\_Ahead ” is when there are open positions (of course they are of same type for the same instrument/symbol) and Symbol\_Net\_Position\_Progressive\_Profit (0) is greater than Symbol\_Net\_Position\_Progressive Profit(1) is greater than Symbol\_Net\_Position\_Profit (2)……. Symbol\_Net\_Position\_Progressive\_Profit (N=3 or 5 or more) and Latest\_Symbol\_Position\_Profit is greater than “Position\_Cost”. (SPA)
5. “Symbol\_Reversal\_Close\_Oldest\_in\_Profit ” is when there are open positions (of course they are of same type for the same instrument/symbol) and Symbol\_Net\_Position\_Profit (0) is lower than sum of “all but the last/latest” Symbol\_positions profits by more than Last/Latest position Cost. (SRCOIP)
6. “Symbol\_Hold\_In\_Loss ” is when there is one open position and Symbol\_Net\_Position\_Profit(0) is negative (Holding a Loss). (SHIL)
7. “Symbol\_in\_Recovery ” is when there are open positions (of course they are of same type for the same instrument/symbol and more than one ) and Symbol\_Oldest\_Position\_Profit is negative. Symbol\_oldest\_Position\_Profit = Symbol\_Net\_Position\_Profit (NOP-1) (SIR)

Below are our External (Input parameters) and Internal (local and used) variables:

External Inputs are satisfied in 1st Section.

Internal local variables defined, used, saved and reported/journaled by the Strategy ( Recapitulation of 1stSection)

1. Boolean MyHedging 🡪 default = False (as US is FIFO), as initialized – but keep for future.
2. Boolean SCS (Symbol\_Clean\_Slate) = default = True, as initialized – checked every tick.
3. Boolean SWA (Symbol\_Walk\_Ahead) = default = False, as initialized – checked every tick.
4. Boolean SPTL (Symbol\_Pause\_To\_Look) = default = False, as initialized – checked every tick.
5. Boolean SSA (Symbol\_Sprint\_Ahead) = default = False, as initialized – checked every tick.
6. Boolean SRCOIP (Symbol\_Reversal\_Close\_Oldest\_in\_Profit) = default = False, as initialized – checked every tick.
7. Boolean SHIL (Symbol\_Hold\_In\_Loss) = default = False, as initialized – checked every tick.
8. Boolean SIR (Symbol\_in\_Recovery) = default = False, as initialized – checked every tick.
9. Integer NOP (Number Open Positions) 🡪 value can be 0 to MaxNOP
10. Table 🡪 **Local Trades Tables** (one per Instrument Symbol) Default Size is (12 x (MaxNOP+6)).  
    Example EURUSD\_LTTable [ 12 Columns x (MaxNOP+6) rows)

The 12 columns are listed below:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Order #  By EA / Server | Order Place Time | Order Exec. Time | Order Place Price | Order Entry Price | Order Position Type | Order Position size | Index of Position | Symbol  Price Now | Order Position Cost | Order Position Profit | Remarks |
| xxxxxxxx |  |  |  |  |  |  | 0 |  |  |  | Oldest |
| xxxxxxxx |  |  |  |  |  |  | 1 |  |  |  |  |
| xxxxxxxx |  |  |  |  |  |  | 2 |  |  |  |  |
| xxxxxxxx |  |  |  |  |  |  | 3 |  |  |  |  |
| xxxxxxxx |  |  |  |  |  |  | 4 |  |  |  | Latest |
| Blank | Blank | Blank | Blank | Blank | Blank | Blank | Blank | Blank | Blank | Blank | Blank Row |
| MaxNOP | 5 |  |  |  |  |  |  |  |  |  |  |
| Blank | Row |  |  |  |  |  |  |  |  |  |  |
| Sum-All | NOP |  |  | Avg Price |  | Total |  |  |  | Total  NetPL | ∑  0 to NOP-1 |
| Sum-all but Last | NOP-1 |  |  | Avg. Price |  | Total  -Last |  |  |  | Sum  NetPL | ∑  0 to (NOP-2) |

Display this table on a fresh Tab/Sub-Window/Area which is like the “Terminal🡪 Trades Tab Window for Symbol

1. SNPP[] & SNPPP[] calculated on every tick.
2. Array double Symbol\_OrderCost [MaxNOP] 🡪 This array is initialized/zeroed at EA “init” and when the EA comes to the state we call “SCS (Symbol\_Clean\_Slate) “; then with open position(s), this array keeps the history of currently open orders/positions – this array is the same column # 10 of the Table – we call Local Trades Table ( para 10 above)- after header, next MaxNop Rows.
3. datetime 🡪 CycleStartTIme, CycleEndTime
4. double 🡪 CycleTime, CycleProfit, CycleCost, AverageCycleTime, AverageCycleProfit, Average CycleCost. WeeeklyProfit, DayEndBalance, DayEndEquity. DayEndProfit.
5. Integer 🡪 RTTradesThisCycle, RTTradesThisWeek

**Rationale, Logic and Goal of the Strategy.**

General.

EA shall run on its own once started/launched. Assume connections with the Trading Server is stable and steady and reliable and not lost. Even with the unlikely event of a disconnection etc. hopefully some standard auto-reconnect feature will be there or at least an alert/mobile or email will be received.

Manual restart / relaunch / etc. are always there as usual.

**Our Logic and our own set of boundaries are described below:**

1. We run on Desktop MT4 Station connected to Trading Server via a reliable broadband internet.  
   After debug test and some time on a DEMO account – we should be ready to put it on a live account on a VPS after seeing the results and gaining confidence on the EA.
2. We preprogram launch time input to start from 5:01 pm EST Sunday to 4:59 pm Friday (New York Time).
3. It mimics as if someone looking at the MT4 Desktop Station rom Sunday 5:10 pm to Friday 4:50 pm.
4. At launch, between 5 and 5:10pm on Sunday it must check that no positions are open and account Equity is equal to Account Balance and margin used is zero and margin available is equal to Equity. So that the SCS (Symbol\_Clean\_Slate) State can be true to start our intended EA trading Cycles for the week. Likewise, after 4:50pm on Friday, it must start an orderly close out of all open trades so that we again reach the same the SCS (Symbol\_Clean\_Slate) State by 5:pm Friday.
5. Active auto- trading will be enabled from 5:10pm EST Sunday to 4:50pm Friday. These are controlled and programmable by the EA Inputs of MTCS, TTOT and TTCT listed on page #2.
6. On pages 1-3, I have listed and enumerated various states, external and internal parameters/variables – by which we shall specify and describe the logic flow to obtain the desired goals. Overall – we enter a position with a logic/expectation, we watch/observe what happens, if in profit we add more according to our logic expecting more profit. These add-more happen during the two states - we call the SWA (Symbol\_Walk\_Ahead) State and the SPA (Symbol\_Sprint\_Ahead) State. There is a State we call SPTL (Symbol\_Pause\_to\_Look) State – when we are in profit, but we are not sure to act either way. There is a State we call SRCOIP (Symbol\_Reversal\_Close\_Oldest\_in\_Profit) State when we close out, respecting USA FIFO rule, the oldest orders holding profit. As a result, we expect to enter the State we call SHIL (Symbol\_Hold\_In\_Loss) State – from where we even may enter the State – we call SIR (Symbol\_in\_Recovery) State or enter any of the other five States. If we enter the SIR (Symbol\_in\_Recovery) State, then we close the Total Net Position. Effect is like SP or SL but individual positions are closed following FIFO rule one at a time from oldest to newest. This, we shall call a “Cycle” – We start with the State **SCS** and we come back to State **SCS**. I shall describe below each State and our logic and actions (trading, watching and reporting and housekeeping etc.).
7. Money, Risk, Leverage, Trade Size Management and use of Preset TP, SL lines are important and those will be clear, as I describe below our Initialization and reset actions and trade and report /log /alert actions in various States.
8. At launch, the external inputs are taken in and are checked for validity too.
   1. During debug – we stay at chosen our defaults as MaxNOP=5 & HMI = 4
   2. We take in the defaults to fill in the required “Instrument Rack” table the arrays called TSM [] and Delta []. Etc.
   3. The EA program checks if the required 4 instruments charts are open, if yes – fine, if not the EA opens the missing Symbol Chart and closes the unwanted ones.
   4. The EA program creates on the required 4 instruments charts (now open) the respective “Local Trades Table” and make them visible like an indicator Window / Tab.
   5. The EA does the EA Initializations and declarations / allocations of all required variables described in pages #2 and #3.
9. We shall run/execute our Expert Function on every tick. (We may skip a tick, if required for a valid reason; or even run the EA on every N tick where N= 1,2,3 as needed using MyLTI, MyDTI and My ATI).

Every tick, we look at symbol positions, if open, count them and fill the various local tables, display arrays we need.

For each Symbol / Instrument, we determine the State (1 of 7) we are in and jump to the corresponding one of seven (a, b, c, d, e, f, g) Symbol State Routines below (next pages #7- #9):

* 1. SCS: Symbol\_Clean\_Slate
     + Look at 3 current (index = 0) candle bars at TF1, TF2, TF3 for their votes. If they are in the same direction as the instrument/symbol “Trade\_Type” inputted in Instrument\_Rack , then the vote is 1 otherwise -1; add the 3 votes and possible total results are +3, +1, -1, -3.
     + If vote is -3, then reverse the “Trade\_Type” and go back to step 10(a);
     + If vote is +3 or +1 then,
       - Enter a Market Entry Order of Type as “Trade\_Type” and size as “MyTradeSize”
       - Set up a good luck TP at ATR (Week,5) from actual Open Price for the trade/position.
       - Set up NO SL. (EA manages own closures and risks when in loss)
       - Gather all the data points of actions and results for corresponding row of the Local\_Trades\_Table for Symbol.
       - Go to Wait for the next pass of the EA.
     + If vote is -1 then,
       - Go to Wait for the next pass of the EA.
  2. SWA: Symbol\_Walk\_Ahead

If NOP >=2 then

* + - * Make sure, for Symbol\_SNPPP(0) > Symbol\_SNPPP(1)
      * OR
      * Current Price is in favorable direction by more than Delta1(NOP) from last position/trade open price
        1. else Go to Wait for the next pass of the EA.
      * Enter a Market Entry Order of Type as “Trade\_Type” and size as “MyTradeSize”
      * No Order Modify to change either TP or SL.
      * Gather all the data points of actions and results for corresponding to this trade/position row of the Local\_Trades\_Table for Symbol.
      * Go to Wait for the next pass of the EA.

If NOP ==1 then

* + - * Current Price is in favorable direction by more than Delta1(NOP) from last position/trade open price
        1. else Go to Wait for the next pass of the EA.
      * Enter a Market Entry Order of Type as “Trade\_Type” and size as “MyTradeSize”
      * No Order Modify to change TP or SL.
      * Gather all the data points of actions and results for corresponding row of the Local\_Trades\_Table for Symbol.
      * Go to Wait for the next pass of the EA.
  1. SPTL: Symbol\_Pause\_To\_Look
     + - If Symbol Oldest Position profit< 0 send and write file Alert “With NOP positions in SPTL with Oldest Position in loss”; Go to Wait for the next pass of the EA
       - If (Symbol Latest Position Profit + Symbol Latest Position Cost x 2 <=0) and (Symbol Oldest Position Profit >Symbol Oldest Position Cost); then Close the Symbol Oldest Position in profit. Gather all the data points of actions and results for the row corresponding to this trade/position in the Local\_Trades\_Table for Symbol.
       - Go to Wait for the next pass of the EA

* 1. SPA: Symbol\_Sprint\_Ahead
     + - Refresh NOP = Number of Symbol Open Positions. // (SPANOP in use).
       - ThisTradeSize= MyTradeSize;
       - If (GREED) the recalculate ThisTradeSize = MyTradeSize x TSM(NOP);
       - Enter a Market Entry Order of Type as “Trade\_Type” and size as “ThisTradeSize”
       - No Order Modify to change TP or SL.
       - Gather all the data points of actions and results including errors and alerts for corresponding row of the Local\_Trades\_Table for Symbol.
       - Go to Wait for the next pass of the EA
  2. SRCOIP: Symbol\_Reversal\_Close\_Oldest\_In\_Profit
     + - Refresh NOP = Number of Symbol Open Positions.
       - If Symbol Oldest Position profit< 0 send and write file Alert “ Holding NOP positions in SRCOIP with Oldest Position in loss”; Go to Wait for the next pass of the EA
       - If (Symbol Latest Position Profit + Symbol Latest Position Cost x 2 <=0) and (Symbol Oldest Position Profit >Symbol Oldest Position Cost); then Close the Symbol Oldest Position in profit. Gather all the data points of actions and results for the row corresponding to this trade/position in the Local\_Trades\_Table for Symbol.
       - Go to Wait for the next pass of the EA
  3. SHIL: Symbol\_Hold\_In\_Loss
     + - Refresh NOP = Number of Symbol Open Positions (must be 1).
       - If (NOP != 1 ) then send and write file Alert “ How with NOP positions in State SHIL”; Go to Wait for the next pass of the EA;
       - If time elapsed since held position entry is more than TF4 then take vote as in SCS with TF1, TF2, TF3; else Go to Wait for the next pass of the EA;
       - If vote = -3 then Close the Held Position in Loss. Gather all the data points of actions and results to be ready for next Cycle with State SCS. Go to Wait for the next pass of the EA;
       - If the vote =3 then calculate ThisTradeSize = Held position size and Enter a Market Entry Order of Type as “Trade\_Type” and size as “ThisTradeSize”
       - No Order Modify to change TP or SL.
       - Gather all the data points of actions and results including errors and alerts for corresponding row of the Local\_Trades\_Table for Symbol.
       - Go to Wait for the next pass of the EA
  4. SIR: Symbol\_In\_Recovery
     + - Refresh NOP = Number of Symbol Open Positions (be >1).// (SIRNOP in use)
       - If (NOP == 1 then send and write file Alert “How with NOP positions in State SIR”; Go to Wait for the next pass of the EA;
       - If (SIR) and if (Symbol\_Net\_Position\_Profit (0) > Symbol\_Net\_Position\_Size \* RecovPoint) then **Close** all held positions per FIFO asap. Gather all the data points of actions and results to be ready for next Cycle with State SCS. Go to Wait for the next pass of the EA;
       - If a New Candle of period TF5, then look at previous 13 candles and if latest consecutive 5 or 8 or 13 candles were all in opposite of Symbol Trade\_Type then recalculate ThisTradeSize = MyTradeSize \* TSM(1 or 2 or 3 respectively); else Go to Wait for the next pass of the EA;
       - Market Entry Order of Type as “Trade\_Type” and size as “ThisTradeSize”
       - No Order Modify to change TP or SL.
       - Gather all the data points of actions and results including errors and alerts for corresponding row of the Local\_Trades\_Table for Symbol.
       - Go to Wait for the next pass of the EA;

1. For each cycle SCS to next SCS capture and file write CycleStartTIme, CycleEndTime
2. For each cycle capture and file write CycleTime, CycleProfit, CycleCost, so that this EA can update after each cycle the AverageCycleTime, AverageCycleProfit, Average CycleCost. Also, the EA calculates and file writes daily and weekly summary reports of the WeeeklyProfit, DayEndBalance, DayEndEquity. DayEndProfit with time stamped rows of the reports
3. EA also captures and file writes RTTradesThisCycle, RTTradesThisWeek (RT stands for Round Trip)