○主題；「オフライン時；任意スプレッドの設定原理」
副題；「history フォルダ中のシステム・ファイルを覗く（その２）」

・アメンボです、

今回は、「history フォルダ中のシステム・ファイルを覗く（その１）」に引き続いて
システム・ファイルの役割をもう少し詳しく調査した結果です。
特に「symbols.sel」ファイルと「バックテスト」の関係に焦点を当てました。

・実は、土日曜の「バックテスト可能化」以外に、かねてから、
ＥＡを評価する一方法として、「スプレッド値を変化させたときの収益変化」を
採用できないかと考えていましたので、設定原理・手順に興味がありました。

・本稿では、「symbols.sel」以外に、バックテスト時に作成される「FXT ファイル」を
調査対象に取り上げています。
色々調べているうちに、また新たな「疑問・不明点」が発生してしまいました、
いったい何時になったら人心地つけるんでしょう！

＜同時掲載資料＞・・ダウンロード用
・アメンボが動作確認に使用したスクリプト一式；
「make_spread_amenbo.zip」を解凍して試してみてください

目次：

１．任意スプレッドの設定手順と結果

※本稿で解説するテクニックは、小生の調べた範囲では既に広く知られている方法だったのですが、
イマイチ原理等が良く判らない部分がありましたので、少し詳細に考察・確認しなおしました。

※MT4 のスプレッド値変更は、オフラインのみで可能です。

折角、任意値に書換えても、オンラインにした途端に更新されて変動スプレッド値に戻ります。

（1）使用するコードの概要

※この章では要点のみを述べます。詳細は「４.、５.」を参照ください。

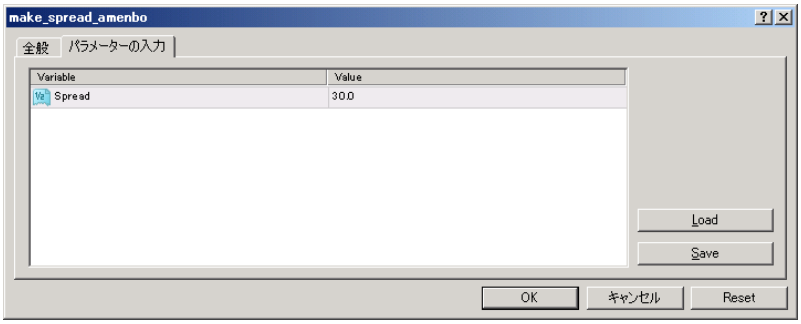
使用 MQL4 コード	タイプ	機能概要
make_spread_amenbo.mq4	スクリプト	・「symbols.sel」中の「スプレッド」値を書換える。 ・書換えた「symbols.sel」は「¥experts¥files」フォルダ内に生成されます。
read_write_symbols_sel.mq4	スクリプト	・「symbols.sel」の読出し・書込み
access_to_FXTfile_02.mq4	スクリプト	・「¥tester¥history」フォルダ中の「FXT ファイル」から「スプレッド」値を読出す
Empty_EA.mq4	ＥＡ	・空っぽのＥＡ（何もしません、ダミーです）

（2）任意スプレッドを設定してみる

＜手順１＞；修正版「symbols.sel」を作成する

・先ず、オンライン状態で、「make_spread_amenbo」をバックテスト対象チャートにアタッチする。

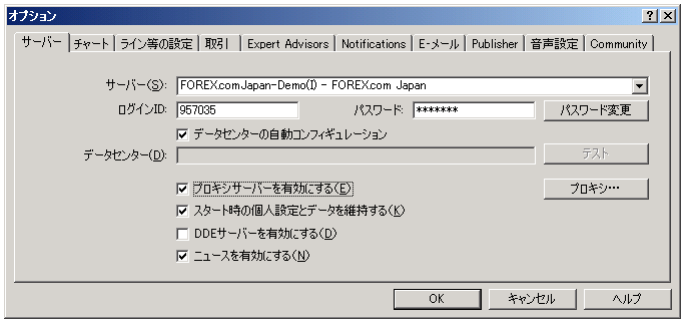
・スプレッド値を設定後、［ＯＫ］

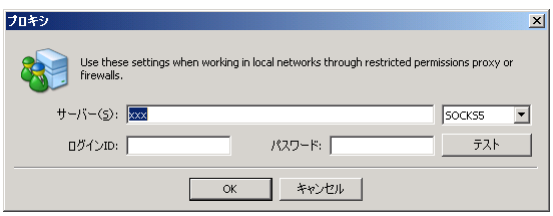


⇒「experts¥files」フォルダ中に、スプレッド値を書換えた「symbols.sel」が作成される。

＜手順２＞；ＭＴ４をオフラインにする

①［プロキシサーバーを有効にする］にチェック、［プロキシ・・］選択し「xxx」入力、［ＯＫ］





⇒　これで、再立上げ後はオフラインになる。

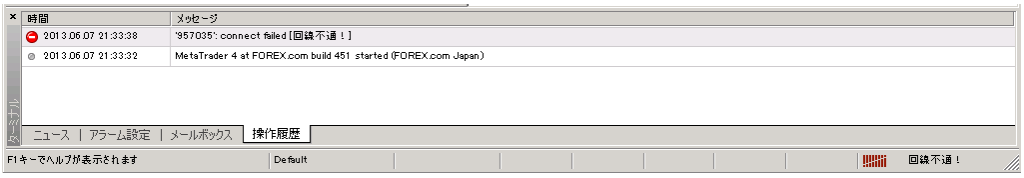
②ＭＴ４を終了する。

＜手順３＞;「symbols.sel」を入れ替える

①「¥experts¥files」フォルダ中に作成された「symbols.sel」を、「¥histroy¥サーバー名」フォルダ内の「symbols.sel」に上書きする。

＜手順４＞;ＭＴ４を再起動する

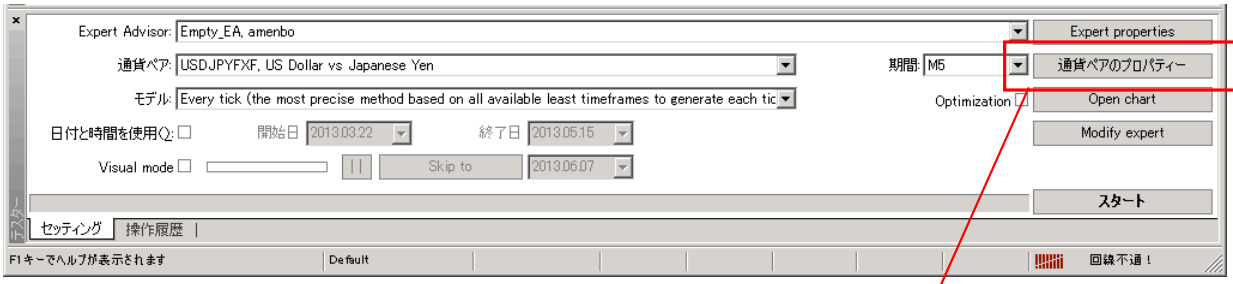
⇒　オフラインであることを確認する。

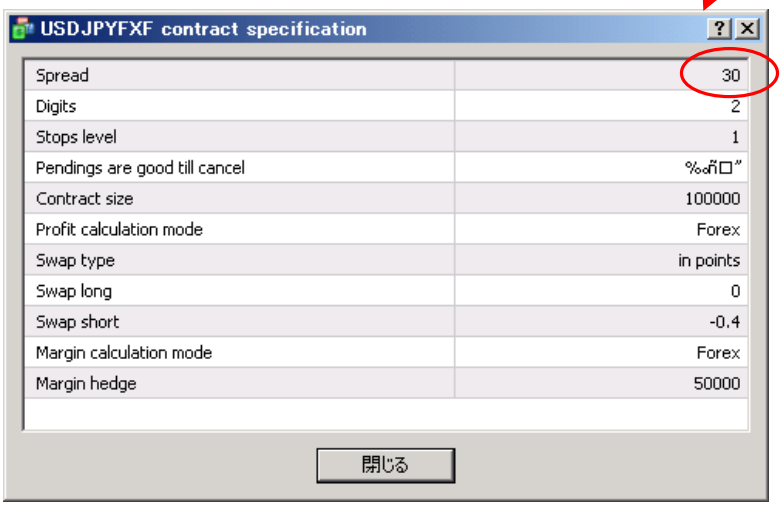


＜手順５＞；スプレッド値を確認する

①Market Watch　ウインドウ　⇒　［USDJPYFXF］のスプレッドが「３０」になった

②バックテスト・タブ中の［通貨ペアのプロパティー］を確認する

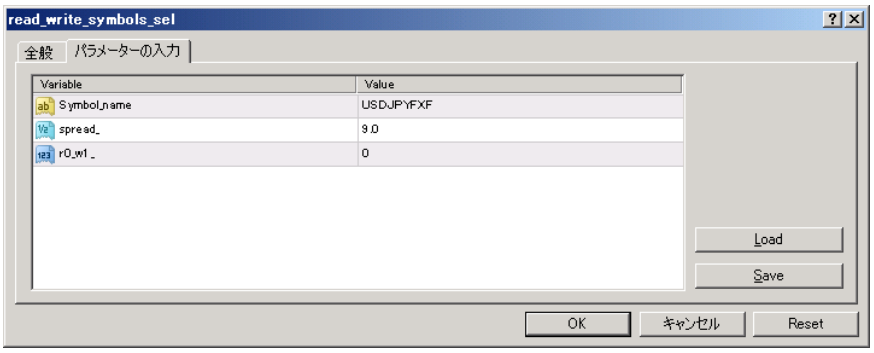




spread=30
になった。

③「¥histroy¥サーバー名」フォルダ中の、「symbols.sel」ファイル内の ASK 値を確認する

読出しには、スクリプト「read_write_symbols_sel.mq4」を使用する。



［r0_w1_=0］とし、［ＯＫ］・・読出しの実施

※「0」が読出し、「0 以外」が書出し

⇒　「¥experts¥logs フォルダ中の「日付.log」ファイル内容」を確認（下記）

```
21:54:35 read_write_symbols_sel USDJPYFXF,M5: loaded successfully
21:54:54 read_write_symbols_sel USDJPYFXF,M5 inputs: Symbol_name="USDJPYFXF"; spread_=9;
r0_w1_=0;
21:54:54 read_write_symbols_sel USDJPYFXF,M5: Version NO= 400
21:54:54 read_write_symbols_sel USDJPYFXF,M5: -------START-------
21:54:54 read_write_symbols_sel USDJPYFXF,M5: ShortName= USDJPYFXF
21:54:54 read_write_symbols_sel USDJPYFXF,M5: Digits= 2
21:54:54 read_write_symbols_sel USDJPYFXF,M5: Point= 0.01
21:54:54 read_write_symbols_sel USDJPYFXF,M5: Time_in_symbols.sel_file:2013 年 6 月 7 日
19 時 17 分 53 秒
21:54:54 read_write_symbols_sel USDJPYFXF,M5: High[0]= 98.64 || Low[0]= 95.54
21:54:54 read_write_symbols_sel USDJPYFXF,M5: Bid_1= 95.72 || ask_1= 96.02
21:54:54 read_write_symbols_sel USDJPYFXF,M5: Bid_2= 95.72 || ask_2= 96.02
21:54:54 read_write_symbols_sel USDJPYFXF,M5: ----------------END---
```

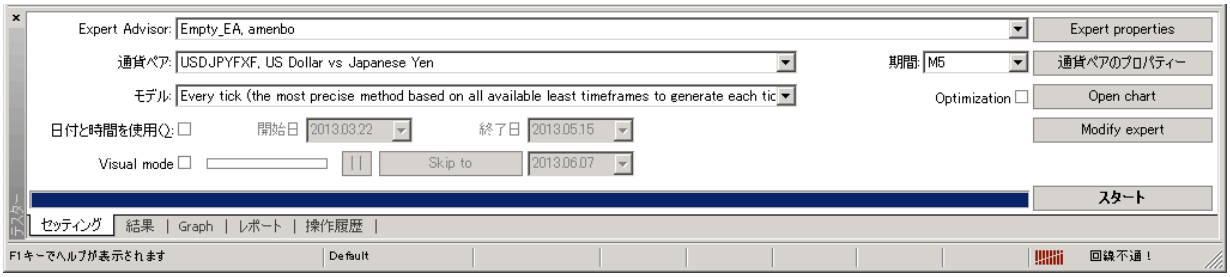
※このスクリプトは、「¥histroy¥サーバー名¥symbols.sel」ファイル内容を
直接に書換える機能を持たせたが、うまく機能していない。（後述）
読込み専用に使っているので、
「inputs: Symbol_name="USDJPYFXF"; spread_=9; r0_w1_=0; 」中の
「spread_=9」は、無視してください。

＜手順６＞・・・バックテストを実施する

実施条件	①ＥＡ	; Empty_EA
	②通貨ペア	; USDJPYFXF
	③モード	; Every tick
	④周期	; M5

［スタート］

⇒　プログレス・バーが、「２回」進行する。



たぶん、

１回目で、「FXT ファイル」が生成され、

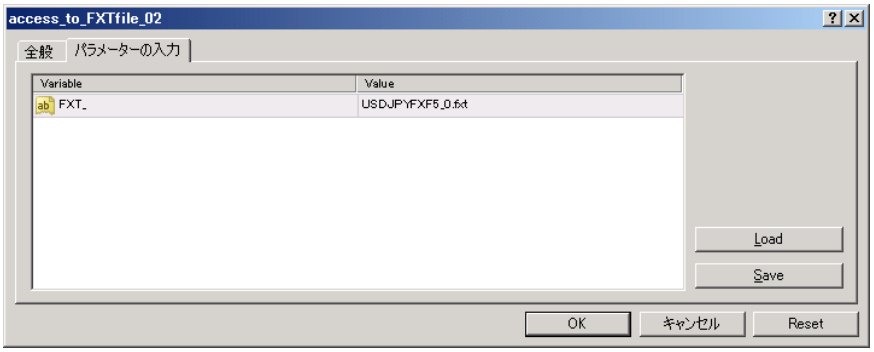
２回目で、上記ファイルを使ったバックテストが行われる。

※アメンボは以前、なぜ「２回」プログレス・バーが進行（伸びる）のかが判らなかった。

⇒　「¥tester¥history」フォルダに「FXT ファイル」が生成される

＜手順７＞・・・FXT ファイルのヘッダーから「スプレッド値」を読み出す

・FXT ファイル「USDJPYFXF5_0.fxt」の内容を、スクリプト「access_to_FXTfile_02.mq4」で読み出してみる



［ＯＫ］

⇒　「¥experts¥logs フォルダ中の「日付.log」ファイル内容」を確認（下記）

```
22:18:27 access_to_FXTfile_02 USDJPYFXF,M5: loaded successfully
22:19:19 access_to_FXTfile_02 USDJPYFXF,M5 inputs: FXT_="USDJPYFXF5_0.fxt";
22:19:19 access_to_FXTfile_02 USDJPYFXF,M5: -------START Win32API-------
22:19:19 access_to_FXTfile_02 USDJPYFXF,M5: win32api_File_Path=
C:¥MT4_back_test¥Forex.com¥tester¥history¥USDJPYFXF5_0.fxt
22:19:19 access_to_FXTfile_02 USDJPYFXF,M5: spread= 30
22:19:19 access_to_FXTfile_02 USDJPYFXF,M5: digits= 2
```

※「symbols.sel」の書き換えで、FXT ファイル内容も変わりました。

２．関連する「バイナリ・ファイル」マップ（地図）と資料

（１）「symbols.sel」マップ再確認

・コードの理解（解読）用として、再度掲載しておきます。

		ｾｸﾀｰ内ｵﾌｾｯﾄ(ﾊﾞｲﾄ)		ﾊﾞｲﾄ数	型	内容	MT4表現型	特記・他(内容)
		10進	16進					1ｾｸﾀｰ＝128(0x80)ﾊﾞｲﾄ
		－	－	4	Int	Version NO	LONG_VALUE	ｾｸﾀｰのｶｳﾝﾄに含めず
第１セクター	1	0	0	12	Char	Symbol Short Name	String	「`¥0`ﾇﾙ」終端
	2	12	0C	4	Int	Digts	LONG_VALUE	
	3	16	10	4	Int	1 とか 263	LONG_VALUE	アメンボ経験値
	4	20	14	4	Int	1	LONG_VALUE	アメンボ経験値
	5			4	Int	4	LONG_VALUE	アメンボ経験値
	6			4	Int	0	LONG_VALUE	(Int)ギャップ
	7	32	**20**	8	Double	Point	DOUBLE_VALUE	
	8	40	28	4	Int	0	LONG_VALUE	(Int)ギャップ
	9	44		4	Int	0	LONG_VALUE	(Int)ギャップ
	10	48	30	4	Int	1	LONG_VALUE	アメンボ経験値
	11	52	34	4	Int	256	LONG_VALUE	アメンボ経験値
	12	56	38	4	Int	TimeCurrent()	LONG_VALUE(datetime)	シャットダウン時のデータ
	13	60	3C	4	Int	0	LONG_VALUE	(Int)ギャップ
	14	64	**40**	8	Double	Bid	DOUBLE_VALUE	シャットダウン時のデータ
	15			8	Double	Ask	DOUBLE_VALUE	シャットダウン時のデータ
	16	80	**50**	8	Double	High[0]	DOUBLE_VALUE	日足データ;シャットダウン時
	17			8	Double	Low[0]	DOUBLE_VALUE	日足データ;シャットダウン時
	18		60	8	Double	0	DOUBLE_VALUE	(Double)ギャップ
	19			8	Double	0	DOUBLE_VALUE	(Double)ギャップ
	20	112	**70**	8	Double	Bid	DOUBLE_VALUE	シャットダウン時のデータ
	21			8	Double	Ask	DOUBLE_VALUE	シャットダウン時のデータ
第２セクター	1	0	0	12	Char	Symbol Short Name	String	「`¥0`ﾇﾙ」終端
	2	12	0C	4	Int	Digts	LONG_VALUE	
	3		10	4	Int	27	LONG_VALUE	
	4			4	Int	1	LONG_VALUE	

・ファイルの先頭には、「４バイト」の「バージョンＮｏ」があります、
その後には、１為替ペアにつき、「**１２８バイト（１６進で８０）**」の記述が続きます。
（　バイナリ・ファイルなので、「文字列」以外はフォーマットが判らないと読み出せません　）

・「アメンボ経験値」とした部分は、内容不明の部分です。

・「**１６進**」で表現すると、重要データのオフセット値が、とても「**切りの良い値**」になります、
「０値」はそのための調整用では？と考えて「(Double)ギャップ」としました。

（2）「FXT ファイル」マップ

※「FXT ファイル」とは、バックテストを実行すると「¥tester¥history」フォルダ中に生成される「USDJPYFXF5_0.fxt」等のことで、データ・モデルにより３種類が作成される。

モデル	生成ファイル例
Every tick	USDJPYFXF5_0.fxt
Control points	USDJPYFXF5_1.fxt
Open prices only	USDJPYFXF5_2.fxt

①アドレス形式・・・ヘッダ部分のみの「アドレスと内容」を記載

FXTファイル・ヘッダー構造体;メモリー・イメージ（その１）
ver. 405　アメンボ推測値　　・アラインメント（パッキング）＝「４」バイト

		メモリ内オフセット		ﾃﾞｰﾀ･ﾊﾞｲﾄ数	C言語型	内容	MT4表現型	備考
		16進	10進	10進				
	1	**0**	0	4	int	virsion	int	
	2	04	4	64	char	copyright[64]	string	「`¥0`ヌル」終端
	3	44	68	??	char	server_name[??]	string	ver.405で追加？
	4	**C0**	192	4		0000		調整用ギャップ
	5	C4	196	12	char	symbol[12]	string	
	6	**D0**	208	4	int	period	int	
	7	D4	212	4	int	model	int	
	8	D8	216	4	int	bars	int	
	9	DC	220	4	time_t	fromdata	datetime	
	10	**E0**	224	4	time_t	todata	datetime	
	11	E4	228	4		0000		調整用ギャップ
	12	E8	232	8	double	modelquality	double	
common parameters	13	**F0**	240	12	char	currency[12]	string	
	14	FC	252	4	int	spread	int	
	15	**100**	256	4	int	digits	int	
	16	104	260	4		0000		調整用ギャップ
	17	108	264	8	double	point	double	
	18	**110**	272	4	int	lot_min	int	
	19	114	276	4	int	lot_max	int	
	20	118	280	4	int	lot_step	int	
	21	11C	284	4	int	stop_level	int	
	22	**120**	288	4	int	gtc_pendings	bool	c++ではbool型あり
profit calcculation parameters	23	124	292	4		0000		調整用ギャップ
	24	128	296	8	double	contract_size	double	
	25	**130**	304	8	double	tick_value	double	
	26	138	312	8	double	tick_size	double	
	27	**140**	320	4	int	profit_mode	int	
swape calcuration	28	144	324	4	int	swap_enable	bool	
	29	148	328	4	int	swap_type	int	
	30	14C	332	4		0000		調整用ギャップ
	31	**150**	336	8	double	swap_long	double	
	32	158	344	8	double	swap_short	double	
	33	**160**	352	4	int	swap_rollover3days	int	
margin calculation	34	164	356	4	int	leverage	int	
	35	168	360	4	int	free_margin_mode	int	
	36	16C	364	4	int	margin_mode	int	
	37	**170**	368	4	int	margin_stopout	int	
	38	174	372	4	int	margin_stopout_mode	int	
	39	178	376	8	double	margin_initial	double	
	40	**180**	384	8	double	margin_maintenance	double	
	41	188	392	8	double	margin_hedge	double	
	42	**190**	400	8	double	margin_divider	double	
	43	198	408	12	char	margin_currency[12]	string	
commisions calculation	44	1A4	420	4		0000		調整用ギャップ
	45	1A8	424	8	double	com_base	double	
	46	**1B0**	432	4	int	com_type	int	
	47	1B4	436	4	int	com_lots	int	
for internal use	48	1B8	440	4	int	from_bar	int	
	49	1BC	444	4	int	to_bar	int	
	50	**1C0**	448	24	int	start_period[6]	int	strat_period[0]～[5]
	51	1D8	472	4	int	set_from	int	
	52	1DC	476	4	int	set_to	int	
	53	**1E0**	480	4	int	freeze_level	int	
	54	1E4	484	244	int	reserved[61]	int	[0]～[60]

②１６進ダンプ・イメージ・・・１６進エディタでダンプした場合の内容イメージ

FXTファイル・ヘッダー構造体;メモリー・イメージ(その2)
ver. 405 アメンボ推測値　　・アラインメント(パッキング)＝「4」バイト

アドレス	00～03	04～07	08～0B	0C～0F	備考
0000	version				
0010	copyright[64]				
0020					
0030					
0040		server_name[??]			server_nameはver.405から？
0050					・バイト数は不明
0060					
0070					
0080					
0090					
00A0					
00B0					
00C0	0000	symbol[12]			
00D0	period	model	bars	fromdate	
00E0	todate	0000	modelquality		
00F0	currency[12]			spread	
0100	digits	0000	point		
0110	lot_min	lot_max	lot_step	stop_level	
0120	gtc_pendings	0000	contract_size		
0130	tick_value		tick_size		
0140	profit_mode	swap_enable	swap_type	0000	
0150	swap_long		swap_short		
0160	swap_rollover3days	leverage	free_margin_mode	margin_mode	
0170	margin_stopout	margin_stopout_mode	margin_initial		
0180	margin_maintenance		margin_hedge		
0190	margin_divider		margin_currency[12]		
01A0	続margin_currency	0000	com_base		
01B0	com_type	com_lots	from_bar	to_bar	
01C0	start_perid[0]	start_perid[1]	start_perid[2]	start_perid[3]	
01D0	start_perid[4]	start_perid[5]	set_from	set_to	
01E0	freeze_level	reserved[0]	reserved[1]	reserved[2]	
01F0	reserved[3]				
0200					
0210					
0220					
0230					
0240		＜ reserved[61] ＞			
0250					
0260					
0270					
0280					
0290					
02A0					
02B0					
02C0				reserved[58]	
02D0	reserved[59]	reserved[60]			
02E0					

（3）「FXT ファイル形式」解読参考資料

※参考資料；

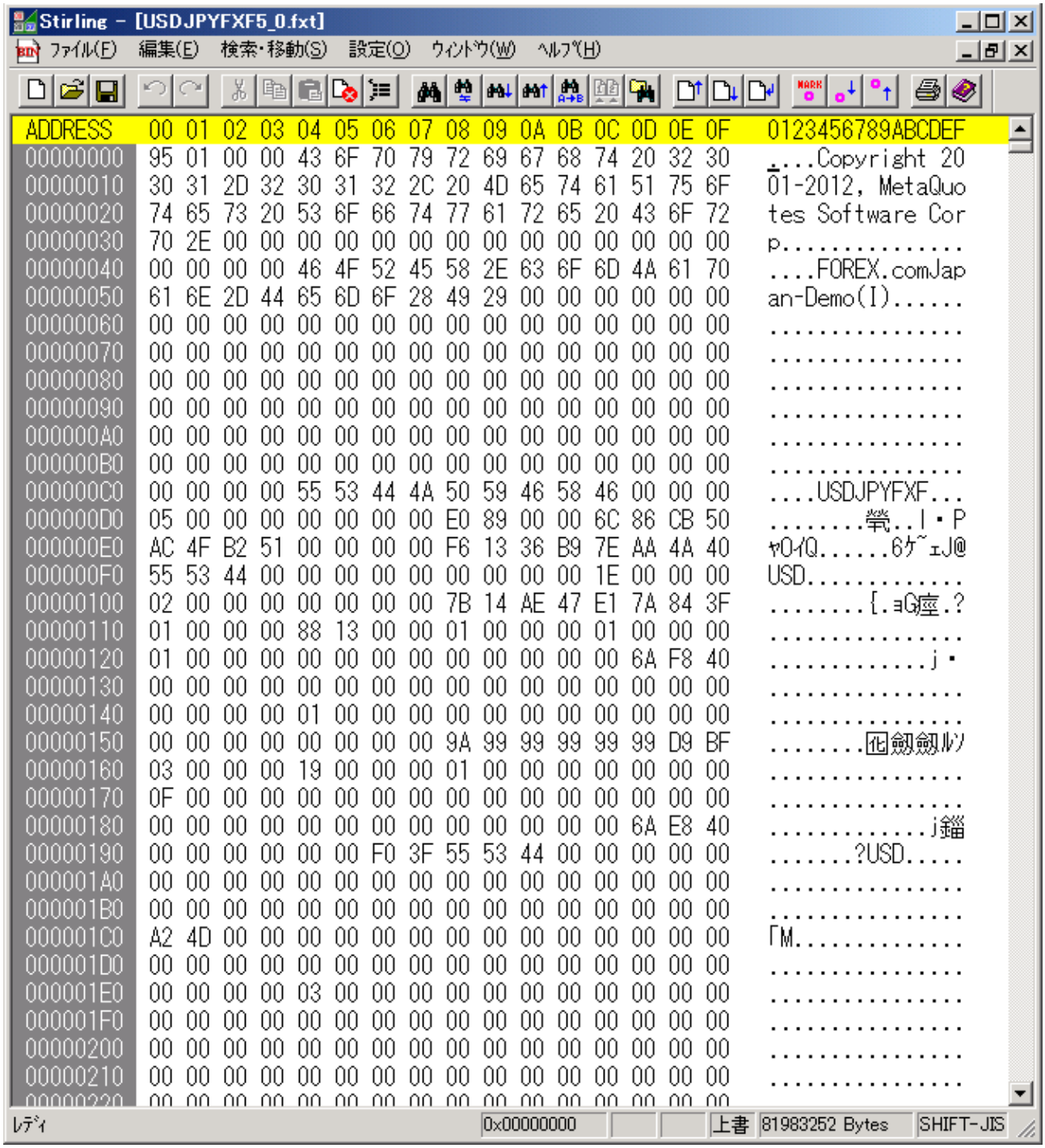
上記「(２)」のマップは、下記「２資料」と、「１６進ダンプ結果」から推測した。

・History Files in FXT Format (version=404)

・FXTHeader.mqh (MetaQuotes Software Corp.) (version=404)

ただし、現状ＭＴ４での「FXT」ファイル・バージョンを調査すると、「version=**405**」で
あったので、内容は「１６進ダンプ結果」との比較で修正しました。(結構、手間が掛かった)

①「１６進ダンプ」結果・・「USDJPYFXF5_0.fxt」



※最初の「４バイト（int)」は「バージョンＮＯ」であろうことは即座に判る。

バージョンＮｏの判別；

・[95 01 00 00]を、数値データはリトル・エンディアンであることから解析すると、

１６進で［0195］となり、更に１０進変換してやると［405］です。

⇒　つまり、「　version=**405**　」と読取れます。

◎アメンボは、結局「version=404」の資料しか入手できなかったので、実際のダンプ結果と、「version=404」資料を比較しながら、解析とマップ作りを進めました。

② History Files in FXT Format (version=404)

History Files in FXT Format

In its operation, tester uses an *.FXT file with generated succession of bars. Each record of the generated succession represents the bar status at either moment within one bar. When modeling bars, tester takes other bars from this file and updates the current bar or adds another one if it has just begun to be formed.
A short description of the format is given below. It begins with the header:

```
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
struct TestHistoryHeader
  {
   int               version;            // 404
   char              copyright[64];      // copyright
   char              symbol[12];
   int               period;
   int               model;              // for what modeling type was
                                 // the ticks sequence generated
   int               bars;               // amount of bars in history
   time_t            fromdate;           // ticks generated from this date
   time_t            todate;             // ticks generating stopped at this date
   double            modelquality;       // modeling quality
   //---- general parameters
   char              currency[12];       // currency base
   int               spread;
   int               digits;
   double            point;
   int               lot_min;            // minimum lot size
   int               lot_max;            // maximum lot size
   int               lot_step;
   int               stops_level;        // stops level value
   int               gtc_pendings;       // instruction to close pending
                                 // orders at the end of day
   //---- profit calculation parameters
   double            contract_size;      // contract size
   double            tick_value;         // value of one tick
   double            tick_size;          // size of one tick
   int               profit_mode;        // profit calculation mode
   { PROFIT_CALC_FOREX, PROFIT_CALC_CFD, PROFIT_CALC_FUTURES }
   //---- swap calculation
   int               swap_enable;        // enable swap
   int               swap_type;          // type of swap
   { SWAP_BY_POINTS, SWAP_BY_DOLLARS, SWAP_BY_INTEREST }
   double            swap_long;
   double            swap_short;         // swap overnight value
   int               swap_rollover3days; // three-days swap rollover
   //---- margin calculation
   int               leverage;           // leverage
```

```
   int               free_margin_mode;   // free margin calculation mode
   { MARGIN_DONT_USE, MARGIN_USE_ALL, MARGIN_USE_PROFIT, MARGIN_USE_LOSS }
   int               margin_mode;        // margin calculation mode
   { MARGIN_CALC_FOREX,MARGIN_CALC_CFD,MARGIN_CALC_FUTURES,MARGIN_CALC_CFDINDEX };
   int               margin_stopout;     // margin stopout level
   int               margin_stopout_mode;// stop out check mode
   { MARGIN_TYPE_PERCENT, MARGIN_TYPE_CURRENCY }
   double            margin_initial;     // margin requirements
   double            margin_maintenance; // margin maintenance requirements
   double            margin_hedged;      // margin requirements for hedged positions
   double            margin_divider;     // margin divider
   char              margin_currency[12];// margin currency
   //---- commission calculation
   double            comm_base;          // basic commission
   int               comm_type;          // basic commission type
   { COMM_TYPE_MONEY, COMM_TYPE_PIPS, COMM_TYPE_PERCENT }
   int               comm_lots;          // commission per lot or per deal
   { COMMISSION_PER_LOT, COMMISSION_PER_DEAL }
   //---- for internal use
   int               from_bar;           // fromdate bar number
   int               to_bar;             // todate bar number
   int               start_period[6];    // number of bar at which the smaller
                                         // period modeling started
   int               set_from;           // begin date from tester settings
   int               set_to;             // end date from tester settings
   //----
   int               freeze_level;       // order's freeze level in points
   //----
   int               reserved[61];
  };
Then, the array of modeled bars follows:
#pragma pack(push,1)
struct TestHistory
  {
   time_t            otm;                // bar time
   double            open;               // OHLCV values
   double            low;
   double            high;
   double            close;
   double            volume;
   time_t            ctm;                // the current time within a bar
   int               flag;               // flag to launch an expert
   (0 - bar will be modified, but the expert will not be launched)
  };
#pragma pack(pop)
```

※上記内容から、「データ配置の順番」と下記の事が判る。

・「ヘッダー部の構造体」と、「データ部の構造体」から構成され、

・「データ部」は、パッキング＝１で、ギュウギュウ詰めにデータ配置されている。

ただし、「ヘッダー部」のパッキング（アラインメント）が正確には判らない！

③ FXTHeader.mqh (MetaQuotes Software Corp.) (version=404)

```
//+------------------------------------------------------------------+
//|                                                   FXTHeader.mqh |
//|                      Copyright ｦ 2006, MetaQuotes Software Corp. |
//|                                        http://www.metaquotes.net |
//+------------------------------------------------------------------+

#define FXT_VERSION         402

//---- profit calculation mode
#define PROFIT_CALC_FOREX     0
#define PROFIT_CALC_CFD       1
#define PROFIT_CALC_FUTURES   2
//---- type of swap
#define SWAP_BY_POINTS        0
#define SWAP_BY_DOLLARS       1
#define SWAP_BY_INTEREST      2
//---- free margin calculation mode
#define MARGIN_DONT_USE       0
#define MARGIN_USE_ALL        1
#define MARGIN_USE_PROFIT     2
#define MARGIN_USE_LOSS       3
//---- margin calculation mode
#define MARGIN_CALC_FOREX     0
#define MARGIN_CALC_CFD       1
#define MARGIN_CALC_FUTURES   2
#define MARGIN_CALC_CFDINDEX  3
//---- basic commission type
#define COMM_TYPE_MONEY       0
#define COMM_TYPE_PIPS        1
#define COMM_TYPE_PERCENT     2
//---- commission per lot or per deal
#define COMMISSION_PER_LOT    0
#define COMMISSION_PER_DEAL   1

//---- FXT file header
int      i_version=FXT_VERSION;                                          //    0 + 4
string   s_copyright="(C)opyright 2006, MetaQuotes Software Corp."; // 64 bytes //   4 + 64
string   s_symbol;                                // 12 bytes              //   68 + 12
int      i_period;                                                        //   80 + 4
int      i_model=0;                               // every tick model     //   84 + 4
int      i_bars=0;                                // bars processed       //   88 + 4
datetime t_fromdate=0;                            // begin modelling date //   92 + 4
datetime t_todate=0;                              // end modelling date   //   96 + 4
//++++ add 4 bytes to align the next double  +++++++
double   d_modelquality=99.0;                                             //  104 + 8
//---- common parameters  -------
string   s_currency;                              // base currency (12 bytes)//  112 + 12
int      i_spread;                                                        //  124 + 4
int      i_digits;                                                        //  128 + 4
//++++ add 4 bytes to align the next double +++++++
double   d_point;                                                         //  136 + 8
int      i_lot_min;                               // minimal lot size     //  144 + 4
int      i_lot_max;                               // maximal lot size     //  148 + 4
int      i_lot_step;                                                      //  152 + 4
int      i_stops_level;                           // stops level value    //  156 + 4
bool     b_gtc_pendings=false;                    // good till cancel     //  160 + 4
//---- profit calculation parameters   -------
//++++ add 4 bytes to align the next double   +++++++
```

```
double   d_contract_size;                                                  //  168 + 8
double   d_tick_value;                                                     //  176 + 8
double   d_tick_size;                                                      //  184 + 8
int      i_profit_mode=PROFIT_CALC_FOREX;         // profit calculation mode  //  192 + 4
//---- swaps calculation   -------
bool     b_swap_enable=true;                                               //  196 + 4
int      i_swap_type=SWAP_BY_POINTS;              // type of swap             //  200 + 4
//++++ add 4 bytes to align the next double   +++++++
double   d_swap_long;                                                      //  208 + 8
double   d_swap_short;                            // overnight swaps values   //  216 + 8
int      i_swap_rollover3days=3;           // number of day of triple swaps  //  224 + 4
//---- margin calculation   -------
int      i_leverage=100;                                                   //  228 + 4
int      i_free_margin_mode=MARGIN_USE_ALL; // free margin calculation mode   //  232 + 4
int      i_margin_mode=MARGIN_CALC_FOREX;         // margin calculation mode  //  236 + 4
int      i_margin_stopout=30;                     // margin stopout level     //  240 + 4
//++++ add 4 bytes to align the next double  +++++++
double   d_margin_initial=0.0;                    // margin requirements      //  248 + 8
double   d_margin_maintenance=0.0;                                         //  256 + 8
double   d_margin_hedged=0.0;                                              //  264 + 8
double   d_margin_divider=1.0;                                             //  272 + 8
string   s_margin_currency;                       // 12 bytes                 //  280 + 12
//---- commissions calculation  -------
//++++ add 4 bytes to align the next double  +++++++
double   d_comm_base=0.0;                         // basic commission         //  296 + 8
int      i_comm_type=COMM_TYPE_MONEY;             // basic commission type    //  304 + 4
int      i_comm_lots=COMMISSION_PER_LOT;    // commission per lot or per deal //  308 + 4
//---- for internal use  -------
int      i_from_bar=0;                            // 'fromdate' bar number    //  312 + 4
int      i_to_bar=0;                              // 'todate' bar number      //  316 + 4
int      i_start_period[6];                                                //  320 + 24
int      i_from=0;                                // must be zero             //  344 + 4
int      i_to=0;                                  // must be zero             //  348 + 4
int      i_reserved[62];                          // unused                   //  352 + 248 =
600

//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void WriteHeader(int handle,string symbol,int period,int start_bar)
  {
//---- FXT file header
   s_symbol=symbol;
   i_period=period;
   i_bars=0;
   s_currency=StringSubstr(s_symbol,0,3);
   i_spread=MarketInfo(s_symbol,MODE_SPREAD);
   i_digits=Digits;
   d_point=Point;
   i_lot_min=MarketInfo(s_symbol,MODE_MINLOT)*100;
   i_lot_max=MarketInfo(s_symbol,MODE_MAXLOT)*100;
   i_lot_step=MarketInfo(s_symbol,MODE_LOTSTEP)*100;
   i_stops_level=MarketInfo(s_symbol,MODE_STOPLEVEL);
   d_contract_size=MarketInfo(s_symbol,MODE_LOTSIZE);
   d_tick_value=MarketInfo(s_symbol,MODE_TICKVALUE);
   d_tick_size=MarketInfo(s_symbol,MODE_TICKSIZE);
   d_swap_long=MarketInfo(s_symbol,MODE_SWAPLONG);
   d_swap_short=MarketInfo(s_symbol,MODE_SWAPSHORT);
   s_margin_currency=StringSubstr(s_symbol,0,3);
   i_from_bar=start_bar;
```

```
   i_start_period[0]=start_bar;
//----
   FileWriteInteger(handle, i_version, LONG_VALUE);
   FileWriteString(handle, s_copyright, 64);
   FileWriteString(handle, s_symbol, 12);
   FileWriteInteger(handle, i_period, LONG_VALUE);
   FileWriteInteger(handle, i_model, LONG_VALUE);
   FileWriteInteger(handle, i_bars, LONG_VALUE);
   FileWriteInteger(handle, t_fromdate, LONG_VALUE);
   FileWriteInteger(handle, t_todate, LONG_VALUE);
   FileWriteInteger(handle, 0, LONG_VALUE);                // alignment to 8 bytes
   FileWriteDouble(handle, d_modelquality, DOUBLE_VALUE);
   FileWriteString(handle, s_currency, 12);
   FileWriteInteger(handle, i_spread, LONG_VALUE);
   FileWriteInteger(handle, i_digits, LONG_VALUE);
   FileWriteInteger(handle, 0, LONG_VALUE);                // alignment to 8 bytes
   FileWriteDouble(handle, d_point, DOUBLE_VALUE);
   FileWriteInteger(handle, i_lot_min, LONG_VALUE);
   FileWriteInteger(handle, i_lot_max, LONG_VALUE);
   FileWriteInteger(handle, i_lot_step, LONG_VALUE);
   FileWriteInteger(handle, i_stops_level, LONG_VALUE);
   FileWriteInteger(handle, b_gtc_pendings, LONG_VALUE);
   FileWriteInteger(handle, 0, LONG_VALUE);                // alignment to 8 bytes
   FileWriteDouble(handle, d_contract_size, DOUBLE_VALUE);
   FileWriteDouble(handle, d_tick_value, DOUBLE_VALUE);
   FileWriteDouble(handle, d_tick_size, DOUBLE_VALUE);
   FileWriteInteger(handle, i_profit_mode, LONG_VALUE);
   FileWriteInteger(handle, b_swap_enable, LONG_VALUE);
   FileWriteInteger(handle, i_swap_type, LONG_VALUE);
   FileWriteInteger(handle, 0, LONG_VALUE);                // alignment to 8 bytes
   FileWriteDouble(handle, d_swap_long, DOUBLE_VALUE);
   FileWriteDouble(handle, d_swap_short, DOUBLE_VALUE);
   FileWriteInteger(handle, i_swap_rollover3days, LONG_VALUE);
   FileWriteInteger(handle, i_leverage, LONG_VALUE);
   FileWriteInteger(handle, i_free_margin_mode, LONG_VALUE);
   FileWriteInteger(handle, i_margin_mode, LONG_VALUE);
   FileWriteInteger(handle, i_margin_stopout, LONG_VALUE);
   FileWriteInteger(handle, 0, LONG_VALUE);                // alignment to 8 bytes
   FileWriteDouble(handle, d_margin_initial, DOUBLE_VALUE);
   FileWriteDouble(handle, d_margin_maintenance, DOUBLE_VALUE);
   FileWriteDouble(handle, d_margin_hedged, DOUBLE_VALUE);
   FileWriteDouble(handle, d_margin_divider, DOUBLE_VALUE);
   FileWriteString(handle, s_margin_currency, 12);
   FileWriteInteger(handle, 0, LONG_VALUE);                // alignment to 8 bytes
   FileWriteDouble(handle, d_comm_base, DOUBLE_VALUE);
   FileWriteInteger(handle, i_comm_type, LONG_VALUE);
   FileWriteInteger(handle, i_comm_lots, LONG_VALUE);
   FileWriteInteger(handle, i_from_bar, LONG_VALUE);
   FileWriteInteger(handle, i_to_bar, LONG_VALUE);
   FileWriteArray(handle, i_start_period, 0, 6);
   FileWriteInteger(handle, i_from, LONG_VALUE);
   FileWriteInteger(handle, i_to, LONG_VALUE);
   FileWriteArray(handle, i_reserved, 0, 62);
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
bool ReadAndCheckHeader(int handle,int period,int& bars)
  {
   int    ivalue;
```

```
   double dvalue;
   string svalue;
//----
   GetLastError();
   FileFlush(handle);
   FileSeek(handle,0,SEEK_SET);
//----
   if(FileReadInteger(handle,LONG_VALUE)!=FXT_VERSION) return(false);
   FileSeek(handle, 64, SEEK_CUR);
   if(FileReadString(handle, 12)!=Symbol())            return(false);
   if(FileReadInteger(handle, LONG_VALUE)!=period)     return(false);
//---- every tick model
   if(FileReadInteger(handle, LONG_VALUE)!=0)          return(false);
//---- bars
   ivalue=FileReadInteger(handle, LONG_VALUE);
   if(ivalue<=0)                                       return(false);
   bars=ivalue;
//---- model quality
   FileSeek(handle, 12, SEEK_CUR);
   dvalue=FileReadDouble(handle, DOUBLE_VALUE);
   if(dvalue<0.0 || dvalue>100.0)                      return(false);
//---- currency
   svalue=FileReadString(handle, 12);
   if(svalue!=StringSubstr(Symbol(),0,3))              return(false);
//---- spread digits and point
   if(FileReadInteger(handle, LONG_VALUE)<0)           return(false);
   if(FileReadInteger(handle, LONG_VALUE)!=Digits)     return(false);
   FileSeek(handle, 4, SEEK_CUR);
   if(FileReadDouble(handle, DOUBLE_VALUE)!=Point)     return(false);
//---- lot min
   if(FileReadInteger(handle, LONG_VALUE)<0)           return(false);
//---- lot max
   if(FileReadInteger(handle, LONG_VALUE)<0)           return(false);
//---- lot step
   if(FileReadInteger(handle, LONG_VALUE)<0)           return(false);
//---- stops level
   if(FileReadInteger(handle, LONG_VALUE)<0)           return(false);
//---- contract size
   FileSeek(handle, 8, SEEK_CUR);
   if(FileReadDouble(handle, DOUBLE_VALUE)<0.0)        return(false);
//---- profit mode
   FileSeek(handle, 16, SEEK_CUR);
   ivalue=FileReadInteger(handle, LONG_VALUE);
   if(ivalue<0 || ivalue>PROFIT_CALC_FUTURES)          return(false);
//---- triple rollovers
   FileSeek(handle, 28, SEEK_CUR);
   ivalue=FileReadInteger(handle, LONG_VALUE);
   if(ivalue<0 || ivalue>6)                            return(false);
//---- leverage
   ivalue=FileReadInteger(handle, LONG_VALUE);
   if(ivalue<=0 || ivalue>500)                         return(false);
//---- unexpected end of file
   if(GetLastError()==4099)                            return(false);
//---- check for stored bars
   if(FileSize(handle)<600+bars*52)                    return(false);
//----
   return(true);
  }
//+------------------------------------------------------------------+
```

※「FXTHeader.mqh」に、「ヘッダー部」のパッキング（アライメント）について重要な記述があり。

＜重要記述＞

所々「double（８バイト)」の前に、下記の記述あり。

「　//++++ add 4 bytes to align the next double +++++++　」

・明らかに「ヘッダー部；構造体」のパッキング（アライメント）を一定値にするための処理です。
当初アメンボは、「パッキング」では、各データ型の最長バイト数に合わせる原則から、
パッキング「８」だと考えたのですが、上記の記述から「４」では？と、考えました。

・「「USDJPYFXF5_0.fxt」」の１６進ダンプ結果は、パッキング「４」と仮定した場合と
良く一致しているので、これを採用してマップを作成しました。

（４）補足；構造体のパッキング（アラインメント）とは

※パッキング（アライメント）とは、データへのアクセス速度を上げる（アクセス回数を減らす）
ための「データの配置処理」のこと。（アライン＝align　整列、位置あわせ）

・32bit CPU では、原則データバスが 32 ビット（＝４バイト）であるので、
一回のメモリ・アクセスで「４バイト」のデータを読み書きすることが出来ます。
つまり、データの先頭を、「４バイト×整数倍」の位置に置く（合わせる）と、データへの
高速アクセスのために都合が良い。（メモリは幾らでも使えると仮定しての話ですが）

（一般にパッキング（アライメント）は２のべき乗（1、2、4、8、16、・・）のどれかを採用します）

・通常、構造体を構成するデータ（下記）の内、最長のバイト数をパッキング（アライメント）と
することが一般的なはずですが、メモリ使用効率も無視できないので決定の際の制約となります。

C 言語	MQL4	パッキング値	備考
char	string	1	文字の長さに係らず「1」とカウント
int	int	4	
double	double	8	

つまり、一般ルールで考えると、FXT ファイル「ヘッダー部；構造体」の、
パッキング（アライメント）＝「８」となります、が、

◎しかし、「**FXT ファイル」では**メモリ使用効率から、パッキング（アライメント）＝「４」とし、
そのための補正用として、所々（ところどころ）double データの直前に「４バイト；int」の
空データ（0000）を「調整用ギャップ」として挿入しているものと、判断しました。

＜パッキング（アライメント）イメージ＞

例えば、データとして「char "AB"」と「int」があったとします。
このときは、パッキング（アライメント)「４」とすると、下記の様にデータが配置されます。

※［¥00］ヌルの後に、１バイトの空（ムダ）が発生しますが、int データは４バイト区切り
の先頭から配置されることになります。（メモリ消費より、アクセス速度を重視）

３．また、判らないことが増えた

・ＭＴ４と言うかMQＬ４では、「一つ判ると、一つ判らないことが増え」てしまいます、
早いとこ理解を完了して「応用（ＥＡ設計法)」や、ＭＴ５へと進みたいと思っているのですが、
しばらくは「ＭＴ４の迷宮」の中で流離いそうです。(～ウーム)

・愚痴はやめて、自身の記憶用メモを兼ねて「新規の疑問」を記録しておきました。

「read_write_symbols_sel.mq4」;

ＭＴ４をオフライン状態にした後、
このスクリプトを実行すると、「インプット」タブで
r0_w1_=0;　とすると、スプレッド値の「読出し」を、
r0_w1_=「0」以外;　とすると、スプレッド値の「書込み」を行います。

？書込みを実施した場合、一瞬はスプレッド値は「９」に変更されます、
ところが、直ぐに正常値（「２」など）に戻ってしまうのです。なぜー？？

４．サンプル・コードについて（若干の解説）

※基本は、コードを読んで頂ければ判るはずなのですが、補足説明をした方が良いと思われる
部分がありますので、まとめておきます。
(と言うか、率直に言いますと、アメンボ自身がつまずいた部分です)

・アメンボの作成するコードでは、見通しを良くするために、敢えて「エラー処理」は殆ど
記載しません。(ご承知おきください)
実用的なコードとするためには、諸兄にて「エラー処理コード」の追加を検討ください。
(追伸；読み易さに重点をおきますので、トリッキーな記述や、汎用性のある記述もしていません)

（１）make_spread_amenbo.mq4

コード；

```
int fh = FileOpenHistory("symbols.sel",FILE_BIN|FILE_READ);
int fs = FileOpen("symbols.sel",FILE_BIN|FILE_WRITE);

int block[];
int arraySize = FileSize(fh)/4;//block[]の各要素は４バイトなので、４で割る


ArrayResize(block,arraySize);
FileReadArray(fh,block,0,arraySize);
FileWriteArray(fs,block,0,arraySize);
```

ポイント；

・上記のコードは、symbols.sel をコピーしているのですが、
その仲介に「int の配列；block[]」を使っています。

・int 配列の各要素は「４バイト」ですから、FileSize2 で取得したデータのサイズ
（バイト数）を、４で割って、必要な配列要素数（配列サイズ）を計算しています。

（２）access_to_FXTfile_02.mq4

コード；

```
#import "kernel32.dll"
   int _lopen(string patg,int of);
   int _lcreat(string path,int attrib);
   int _llseek(int handle,int offset,int origin);
   int _lread(int handle,int& buffer[],int bytes);
   int _lwrite(int handle,string buffer,int bytes);
   int _lclose(int handle);
#import

      int buffer[2];
      //
      string path=TerminalPath()+"¥¥tester¥¥history¥¥"+FXT_;
      //
      Print("win32api_File_Path= ",path);
      //
      int fileHandle=_lopen(path,0);
      //-----------------------------------------------
      int adjustCursor1=_llseek(fileHandle,0xfc,0);//16進表示
      int ret_read1=_lread(fileHandle,buffer,8);
      Print("spread= ",buffer[0]);
      Print("digits= ",buffer[1]);
```

ポイント；

・C言語で使われる「 _lread()」では、下記の様に解説されています。

_lread(ファイルへのハンドル、データを収納するバッファへのポインタ、読込むバイト数)

つまり、２番目の引数は「ポインタ」です。

読込む対象によって下記の様に書換えればＯＫです。

「_lread()」自体は読込むデータの型なぞ関係なく、バイナリ・データを読込んでいるだけ！

バッファに格納された**データの型を解釈するのは、使う側**なのです。

読込むデータが		コード組合せ
文字列ならば	→	string buffer="1234567890"; _lread(int handle,string buffer,int bytes);
int 型ならば	→	int buffer[]; _lread(int handle,int& buffer[],int bytes);
double 型ならば	→	double buffer[]; _lread(int handle,double& buffer[],int bytes);

・アメンボは試していませんが、「_lwrite()」も同じことだと思います。

・また、既にお判りの様に下記のコードでは動きません。（念のため）

```
int buffer;
_lread(int handle,int& buffer,int bytes);
```

配列の読み書きは、ポインタ経由になっているのですが、

一方、数値データはダイレクト・アクセスで、（たぶん）スタック？経由のため。（違った？）

５．使用コード一覧

（１）make_spread_amenbo.mq4

```
//+------------------------------------------------------------------+
//|                                              make_spread_amenbo.mq4 |
//|                                                              amenbo |
//|                                                  泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"

#property show_inputs

extern double Spread = 30.0;

int start()
{
  double pip = Point;

  int fh = FileOpenHistory("symbols.sel",FILE_BIN|FILE_READ);
  int fs = FileOpen("symbols.sel",FILE_BIN|FILE_WRITE);

  int block[];
  int arraySize = FileSize(fh)/4;//block[]の各要素は４バイトなので、４で割る

  ArrayResize(block,arraySize);
  FileReadArray(fh,block,0,arraySize);
  FileWriteArray(fs,block,0,arraySize);

  FileClose(fs);
  FileClose(fh);
  //----------------------------------------
  fs = FileOpen("symbols.sel",FILE_BIN|FILE_READ|FILE_WRITE);
  string temp, symbol;
  double bid, ask;

  FileSeek(fs,4,SEEK_SET);

  while(!IsStopped()) {
          temp = FileReadString(fs,12);
          if(FileIsEnding(fs)) break;
          symbol = StringSubstr(temp,0,StringFind(temp,"¥x00",0));

          if(symbol==Symbol()){
                  int pos = FileTell(fs)-12;
                  FileSeek(fs,pos+0x40,SEEK_SET);
                  bid = FileReadDouble(fs,DOUBLE_VALUE);
                  ask = NormalizeDouble(bid+Spread*pip,Digits);
                  FileSeek(fs,pos+0x40+DOUBLE_VALUE,SEEK_SET);
                  FileWriteDouble(fs,ask,DOUBLE_VALUE);
                  FileSeek(fs,pos+0x70+DOUBLE_VALUE,SEEK_SET);
                  FileWriteDouble(fs,ask,DOUBLE_VALUE);
                  break;
          }

          FileSeek(fs,0x80-12,SEEK_CUR);
  }
  FileClose(fs);

  PlaySound("alert2.wav");

  return(0);
}
```

(2) read_write_symbols_sel.mq4

```
//+------------------------------------------------------------------+
//|                                      read_write_symbols_sel.mq4 |
//|                                                          amenbo |
//|                                                泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"

#property show_inputs
//
extern string Symbol_name="USDJPYFXF";//デフォルトの為替ペア
extern double spread_=9.0;
extern int r0_w1_=0;       //0=read、1=wriet
//
int sector=0x80;//=128  １為替ペアのレコード・サイズ
int version_offset=4;//4ﾊﾞｲﾄ  ファイルの先頭に４バイトのバージョン情報あり
//
int start()
  {
   //
   int hFile = FileOpenHistory("symbols.sel", FILE_BIN|FILE_READ|FILE_WRITE);
   int symbols_number=(FileSize(hFile)-4)/sector;//いくつの為替ペア情報があるか
   //
   FileSeek(hFile,0, SEEK_SET);
   string version=FileReadInteger(hFile,LONG_VALUE);
   Print("Version NO= ",version);
   //
   FileSeek(hFile,version_offset, SEEK_SET);
   //
   for(int i=0;i<symbols_number;i++)
   {
   //
      int offset=FileTell(hFile);
      string short_name=FileReadString(hFile, 12);
      //
      if(short_name==Symbol_name)
      {
         //
         Print("-------START-------");
         //
         Print("ShortName= ",short_name);
         //
         FileSeek(hFile,(offset+0x0c), SEEK_SET);
         int digits_=FileReadInteger(hFile, LONG_VALUE);
         Print("Digits= ",digits_);
         //
         FileSeek(hFile,(offset+0x20), SEEK_SET);
         double point_=FileReadDouble(hFile, DOUBLE_VALUE);
         Print("Point= ",point_);
         //
         FileSeek(hFile,(offset+0x38), SEEK_SET);
         datetime time_current=FileReadInteger(hFile, LONG_VALUE);
         //Print("TimeCurrent() in symbols.sel_file= ",time_current);
         //
         int year_=TimeYear(time_current);
         int month_=TimeMonth(time_current);
         int day_=TimeDay(time_current);
         int hour_=TimeHour(time_current);
         int minute_=TimeMinute(time_current);
```

```
         int second_=TimeSeconds(time_current);
         Print("Time_in_symbols.sel_file:",year_,"年",month_,"月",day_,"日",hour_,"時
",minute_,"分",second_,"秒");
         //-------------------------------------
         FileSeek(hFile,(offset+0x50), SEEK_SET);
         double high_0=FileReadDouble(hFile, DOUBLE_VALUE);
         double low_0=FileReadDouble(hFile, DOUBLE_VALUE);
         Print("High[0]= ",high_0," || Low[0]= ",low_0);
         //-------------------------------------
         //
         if(r0_w1_==0)
         {
         //
         FileSeek(hFile,(offset+0x40), SEEK_SET);
         double bid_1=FileReadDouble(hFile, DOUBLE_VALUE);
         double ask_1=FileReadDouble(hFile, DOUBLE_VALUE);
         Print("Bid_1= ",bid_1," || ask_1= ",ask_1);
         //
         Sleep(100);
         //
         FileSeek(hFile,(offset+0x70), SEEK_SET);
         double bid_2=FileReadDouble(hFile, DOUBLE_VALUE);
         double ask_2=FileReadDouble(hFile, DOUBLE_VALUE);
         Print("Bid_2= ",bid_2," || ask_2= ",ask_2);
         //
         }else{
         //
         FileSeek(hFile,(offset+0x40), SEEK_SET);
         double bid_3=FileReadDouble(hFile, DOUBLE_VALUE);
         double ask_3=NormalizeDouble(bid_3+spread_*point_,digits_);
         FileSeek(hFile,(offset+0x48), SEEK_SET);
         FileWriteDouble(hFile,ask_3,DOUBLE_VALUE);
         FileFlush(hFile);
         //
         Sleep(100);
         //
         FileSeek(hFile,(offset+0x70), SEEK_SET);
         double bid_4=FileReadDouble(hFile, DOUBLE_VALUE);
         double ask_4=NormalizeDouble(bid_4+spread_*point_,digits_);
         FileSeek(hFile,(offset+0x78), SEEK_SET);
         FileWriteDouble(hFile,ask_4,DOUBLE_VALUE);
         FileFlush(hFile);
         //
         Sleep(100);
         //
         }
         //-------------------------------------
         Print("---------------END---");
         //
         break;
      }
      FileSeek(hFile,(sector-12),SEEK_CUR);

   }

   FileClose(hFile);

   PlaySound("alert2.wav");
   return(0);
}
//--------------
```

(３) access_to_FXTfile_02.mq4

```
//+------------------------------------------------------------------+
//|                                        access_to_FXTfile_02.mq4 |
//|                                                          amenbo |
//|                                                泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"

#property show_inputs

#import "kernel32.dll"
   int _lopen(string patg,int of);
   int _lcreat(string path,int attrib);
   int _llseek(int handle,int offset,int origin);
   int _lread(int handle,int& buffer[],int bytes);
   int _lwrite(int handle,string buffer,int bytes);
   int _lclose(int handle);
#import

extern string FXT_="USDJPYFXF5_0.fxt";

int start()
  {
      Print("-------START Win32API-------");
      //
      int buffer[2];
      //
      string path=TerminalPath()+"¥¥tester¥¥history¥¥"+FXT_;
      //
      Print("win32api_File_Path= ",path);
      //
      int fileHandle=_lopen(path,0);
      //-----------------------------------------------
      int adjustCursor1=_llseek(fileHandle,0xfc,0);//16進表示
      int ret_read1=_lread(fileHandle,buffer,8);
      Print("spread= ",buffer[0]);
      Print("digits= ",buffer[1]);
      //-----------------------------------------------
      int ret_close=_lclose(fileHandle);
   //
   PlaySound("alert2.wav");
   //
   return(0);
}
```

（４）Empty_EA.mq4

```
//+------------------------------------------------------------------+
//|                                                   Empty_EA.mq4 |
//|                                                         amenbo |
//|                                                泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"
//
extern int a=10;
extern double b=3.0;

//
int init()
  {

   return(0);
  }
//
int deinit()
  {

   return(0);
  }
//-----------------------------------
int start()
  {

   return(0);
  }
//-----------------------------------
```

以　上