



トレードステーション



プログラム開発

ひでぼー

目次

第1章 インジケータ作成

- 1 EasyLanguage (EL) の構成-語、式、句読点、文
- 2 発行済株数・時価総額
- 3 発行済株数・時価総額 (2)
- 4 発行済株数・時価総額 (3)
- 5 一目均衡表・改 (基準線・転換線の乖離率追加)
- 6 PEGレシオ
- 7 PEGレシオ (2)
- 8 グレアムのミックス係数
- 9 グレアムのミックス係数 (2)
- 10 エクセル読み書き
- 11 エクセル読み書き (2)
- 12 連結/単独問題
- 13 連結/単独問題 (2)
- 14 ファンダメンタルデータ
- 15 ファンダメンタルデータ (2)
- 16 ファンダメンタルデータ (3)
- 17 投資の女神様
- 18 営業利益・進捗
- 19 営業利益・進捗 (2)
- 20 営業利益・進捗 (3)
- 21 ボラティリティ
- 22 基準日変動率%
- 23 外部データCSV (読込)
- 24 外部データCSV (読込) (2)
- 25 決算発表日
- 26 IPO初値
- 27 IPO公募
- 28 IPO公募 (2)
- 29 ファンダメンタルデータ (検証用)
- 30 ファンダメンタルデータ (検証用) (2)

第2章 アプリ作成

- 1 データの共有
- 2 サンプルプログラム
- 3 サンプルプログラム (2)
- 4 サンプルプログラム (3)
- 5 サンプルプログラム (4)
- 6 サンプルプログラム (5)
- 7 アプリの作り方
- 8 シンボルリンク
- 9 業績推移
- 10 業績推移 (2)
- 11 業績推移 (3)
- 12 業績推移 (4)
- 13 業績推移 (5)
- 14 業績推移 (6)
- 15 業績推移 (7)
- 16 業績推移 (8)
- 17 業績推移 (9)
- 18 業績推移 (10)
- 19 業績推移 (11)
- 20 業績推移 (12)
- 21 業績推移 (13)
- 22 業績推移 (14)
- 23 業績推移 (15)

EasyLanguage (EL) の構成 — 語、式、句読点、文

● 語 — 5種類、語だけに!(°_°) EasyLanguage (以後EL)

- ・予約語：ELで事前に定義された言語 Open (始値)、Close (終値)、High (高値)、Low (安値) 等
- ・関数：ELで定義済みの式を呼ぶ語 エクセルなどにもある関数と同じ感じですAverage、RSI等
- ・ユーザ定義語：ユーザーが任意に定義した語 Input、Var等で宣言してから使います
- ・スキップ語：ELの命令実行時に認識されない語 可読性を上げる為用、Of、The、At等
- ・属性：インジケーターやストラテジーの動作、計算ルール等を設定する語 LegacyColorValue等

● 式 — 2種類

- ・数値式：数値値の参照や計算
- ・真偽式：2つの値を比較して真か偽か判断

● 句読点 — 8種類

- ； — セミコロン、文の末尾に使います
- () — 丸括弧、算術演算子グループ化に使います
- , — コンマ、リスト内のアイテムを区切ります
- [] — 角括弧、過去の参照、プロット表示、配列要素の参照等に使います
- ” ” — 複引用符、テキスト又はラベル等に使います
- : — コロン、宣言に使います
- { } — 波括弧、記述やコメントに使います
- // — Wスラッシュ、記述やコメントに使います

● 文 — 大きく分けて3種類、語、式、句読点等を使ってルール通りに文を構成します

- ・宣言文：システムやクラスの参照、インプット、変数の宣言等
- ・命令文：実質のプログラム 数式計算、If-endの比較判断、While-endの繰り返し等
- ・実行文：画面への表示（プロット：インジケーター用）、売買の実行（売買：ストラテジー用）等

● 文の色

- ・黒色：定義済み変数
- ・灰色：属性
- ・紫色：関数
- ・青色：予約語、引用
- ・緑色：コメント、スキップ語
- ・茶色：文字列（公式は濃い赤、この本では見た目茶色で公式の濃い赤と同じ色）

● 文の順序：一般的なガイドライン

- ・システム、クラスの参照
 - ・インプット宣言、変数宣言
 - ・命令文、実行文
- ※ ELは大文字と小文字を区別しません、可読性向上の改行、スペース、段落は無視します

発行済株数・時価総額

レーダースクリーン用に新しく任意のインジケータを作成します

新規 インジケータ

名前(N): 株式(発行済株数・時価総額) 省略名(H): 株 式

注記(S): 発行済株数・時価総額

次のウインドウに適用する(V):

- チャート分析
- レーダースクリーン

テンプレートを選択(T): (None)

OK キャンセル ヘルプ

● プログラム

```
Value1 = GetFundData("CAC_OUTS2", 0); // NOMURA 400 タイプの株式発行高数  
Plot1(Value1 / 1000, !("発行済数 (千)")); // 発行済株数計算 (発行済株数/1千)  
Plot2(Close * Value1 / 100000000, !("時価総額 (億)")); // 時価総額計算 (終値*発行済株数/1億)
```

● 解説

GetFundData (読取るデータ名, 過去のデータ期間数 (大きい程過去))
Value1 に最新の期間のファンダメンタルデータ「CAC_OUTS2」(発行済株数)の数値を代入します
Plot1にValue1を1000株単位にまとめて代入します
Plot2にClose (終値の予約語) × 発行済株数を掛けて1億単位にまとめて代入します
"発行済数 (千)", "時価総額 (億)" は見出しとして表示されます
Plot1, Plot2は画面に項目を表示する予約語です
このプログラムの場合、Plot1に計算された発行済株数、Plot2に計算された時価総額が表示されます

● 詳細

Value1は数値変数です
数値変数は、計算の数値結果を保存するために使用されます、値を格納する箱みたいなものです
独自の変数は宣言が必要ですが、Value0からValue99は100個事前宣言されていて宣言なしで使用可能
変数は他に、文字列変数 (宣言必要)、真/偽変数 (事前宣言済みCondition0からCondition99の100個) 3種類があります
数値タイプは更に3種類のサブタイプがあります

- ・ 整数 - 32 ビット符号付き整数 (整数のみ)
- ・ 単フロート - 4 バイトとして表される実数、浮動小数点
- ・ ダブルフロート - 8 バイトとして表される実数、浮動小数点

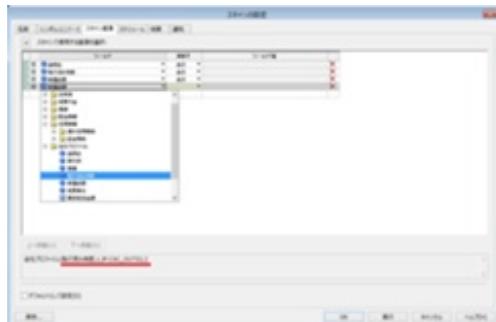
(大きな数字を計算するにはこれで精度が最高となり、丸めによる不一致の可能性を低くします)

タイプ	予約語	開始	終了
整数	int	-2147483648	2147483647
単フロート	float	3.4E -38 (7桁)	3.4E +38 (7桁)
ダブルフロート	double	1.7E -308 (15桁)	1.7E +308 (15桁)

特定のタイプを宣言しない場合、検証時に変数タイプを自動検出して設定します
事前定義済みの変数 (Value0-99) は、ダブルに初期設定されます

発行済株数・時価総額 (2)

スキャナーでもこのフィールドが使用されているのが分かります



作成したインジケータを「分析テクニックを挿入」で選択すると、レーダースクリーンに発行済株数と時価総額が表示されます

	銘柄コード	銘柄名	株 式	
			発行済数(千)	時価総額(億) ▼
1	3563	スシローグローバルホールディングス	27,459	953
2	3978	マクロミル	38,622	731
3	3479	ティーケーピー	4,730	608
4	3964	オークネット	26,269	336
5	3561	力の源ホールディングス	11,250	314
6	9519	レノバ	18,383	313
7	4597	ソレイジア・ファーマ	84,046	212
8	6547	グリーンズ	12,000	174
9	3983	オロ	4,000	157
10	9325	ファイズ	2,523	154
11	3981	ピーグリー	5,879	140
12	3979	うるる	3,050	120
13	3558	ロコンド	5,170	112
14	6544	ジャパンエレベーターサービスホールディングス	9,580	110
15	6175	ネットマーケティング	6,772	108
16	3560	ほほ日	2,250	99
17	3557	ユナイテッド&コレクティブ	1,367	72
18	6545	インターネットインフィニティ	1,254	68
19	3976	シャノン	1,379	61
20	3559	ピーバンドットコム	2,190	55
21	3562	No. 1	1,469	52
22	6546	フルテック	5,200	51
23	6543	日宣	1,938	48
24	6694	ズーム	2,208	41
25	1439	安江工務店	1,299	22
26	3977	フュージョン	720	12
27	3984			
28				

TradeStationで作成。©TradeStation Technologies, Inc. All rights reserved./無断複写・転載を禁じます。

発行済株数・時価総額 (3)

株式（発行済株数・時価総額）をバージョンアップしてみました

浮動株比率を入力して、浮動株比率%×発行済株数÷100で浮動株数を、出来高÷浮動株数×100で売買回転率を算出します

浮動株比率 = (公募数+売出数) ÷ (発行済株数+OA数) で計算して入力しています（実際にOAされるまで数%誤差があります）

業績、割高関係なく、出来高が多く高回転率なほどボラが大きいのがよくわかります

出来高が増えても上がらなくなったり下がってる場合は、天井又は逆回転のサインなので要注意です(°_°)

	銘柄コード	銘柄名	株 式					出来高 (千)	メモ	現	前日比%	前比
			発行済株数 (千)	時価総額 (億)	浮動 比%	浮動株数 (千)	回転 率%					
1	4597	ソレイジア・ファーマ	84,046	334	22.4	18,826	207.04	38,978		397	1.53%	6
2	6175	ネットマーケティング	6,772	88	13.9	941	152.28	1,433		1,300	15.76%	177
3	9325	ファイズ	2,523	141	21.8	550	85.56	471		5,600	-15.15%	-1,000
4	9519	レノバ	18,383	231	6.6	1,213	38.87	472		1,258	-3.31%	-43
5	3985	テモナ	1,275	67	21.9	279	38.46	107		5,280	-10.51%	-620
6	3479	ティーケーピー	4,730	474	10.6	501	36.44	183		10,030	-4.02%	-420
7	3976	シャノン	1,379	44	10.9	150	21.96	33		3,205	-7.37%	-255
8	6545	インターネットインフィニティ	1,254	49	15.5	194	20.98	41		3,930	-4.38%	-180
9	3561	力の源ホールディングス	11,250	224	8.9	1,001	13.40	134		1,989	2.26%	44
10	3558	ロコンド	5,407	91	29.1	1,573	12.06	190		1,680	-0.06%	-1
11	3981	ビーグリー	5,879	101	73.3	4,309	10.41	449		1,717	-5.61%	-102
12	3562	No. 1	1,469	36	28.7	422	8.75	37		2,455	-2.46%	-62
13	6694	ズーム	2,208	32	27.4	605	6.58	40		1,446	-3.60%	-54
14	3557	ユナイテッド&コレクティブ	1,367	59	18.9	258	5.65	15		4,300	-2.27%	-100
15	3979	うるる	3,241	99	39.4	1,277	5.39	69		3,040	-2.41%	-75
16	3983	オロ	4,000	106	24.1	964	4.75	46		2,660	-3.27%	-90
17	3560	ほぼ日	2,250	95	17.3	389	4.73	18		4,235	0.12%	5
18	6546	フルテック	5,368	46	20.9	1,122	4.14	47		853	-0.81%	-7
19	3564	LIXILピパ	44,720	927	41.7	18,648	3.99	745		2,072	0.19%	4
20	1439	安江工務店	1,299	20	37.8	491	3.64	18		1,555	-3.42%	-55
21	3984	ユーザーローカル	3,618	287	87.0	3,148	2.90	91		7,920	-5.71%	-480
22	6544	ジャパンエレベーターサービスホール	10,015	102	29.0	2,904	2.59	75		1,021	-3.68%	-39
23	3559	ピーバンドットコム	2,190	45	35.6	780	2.46	19		2,053	-4.51%	-97
24	6547	グリーンズ	12,000	157	34.8	4,176	2.25	94		1,309	1.08%	14
25	3964	オークネット	26,269	339	17.9	4,702	2.17	102		1,290	-2.64%	-35
26	6543	日宣	1,938	45	12.9	250	1.72	4		2,310	-1.07%	-25
27	3977	フュージョン	720	10	22.2	160	1.69	3		1,435	-2.71%	-40
28	3978	マクロミル	38,622	740	63.1	24,371	1.32	321		1,915	-1.24%	-24
29	3563	スシローグローバルホールディングス	27,459	973	64.9	17,821	0.57	101		3,545	-0.28%	-10
30	7940	ウェブロックホールディングス	0	0	44.3	0		381		632	-4.39%	-29

TradeStationで作成。©TradeStation Technologies, Inc. All rights reserved/無断複写・転載を禁じます。

inputs: Hudou(100)(DisplayName = "浮動株比率%", ToolTip = "浮動株比率%");

```

Value1 = GetFundData("CAC_OUTS2", 0); // NOMURA 400 タイプの株式発行高数
if Value1 = 0 then Value1 = GetFundData("CAB_SHARES_AFTER_TRANSFER", 0); // 名義書換後の株数
Value2 = ( Volume[0] / 1000 ); // 当日出来高/1000

Plot1( Value1 / 1000, !( "発行済株数 (千)" )); // 発行済株数計算 (発行済株数/1千)
Plot2( Close * Value1 / 100000000, !( "時価総額 (億)" )); // 時価総額計算 (終値*発行済株数/1億)
Plot3( Hudou, !( "浮動比%" )); // 浮動株比率%
Plot4( Hudou * Plot1 / 100, !( "浮動株数 (千)" )); // 浮動株数計算 (浮動株比率%*発行済株数/100)
If Plot4 < 0 then Plot5( Value2 / Plot4 * 100, !( "回転率%" )); // 売買回転率計算 (当日出来高/浮動株数*100)
Plot6( Value2, !( "出来高 (千)" )); // 当日出来高数 (千)

```

※ 7940 ウェブロックホールディングスは元データが入っていないため現在表示されていないようです

※ いつまでもデータ入らないので調べたら名義書換後の株数の所にそれっぽい数字があったのでゼロの場合そちらを入れる様に変更

一目均衡表・改（基準線・転換線の乖離率追加）

新しく任意のインジケータを作成して既存の一目均衡表のインジケータをまるっとコピーします

182、183行辺りで計算した転換線、基準線をPlot1、Plot2に代入していますので、そのすぐ後ろに下のプログラム2行を追加します。既にPlot10まで使用しているためPlot11以降を使用します

```

181
182 Plot1( Tenkan_Sen, !( "転換" ) );
183 Plot2( Kijun_Sen, !( "基準" ) );
184
185 Plot11( (( Close - Tenkan_Sen ) / Tenkan_Sen ), !( "転換乖離" ) ); // 追加部分 転換線乖離
186 Plot12( (( Close - Kijun_Sen ) / Kijun_Sen ), !( "基準乖離" ) ); // 追加部分 基準線乖離
187
188 { if in a chart, plot lines and cloud; if in a grid application,
189   plot appropriate text in cells and color cell backgrounds }

```

- プログラム（事前宣言されたものばかり使用するので宣言は要らず、この計算2行だけでOKです）

Plot11(((Close - Tenkan_Sen) / Tenkan_Sen), !("転換乖離")); // 追加部分 転換線乖離

Plot12(((Close - Kijun_Sen) / Kijun_Sen), !("基準乖離")); // 追加部分 基準線乖離

項目並べ替えますが既存一目均衡表との比較です

銘柄コード	銘柄名	一目					現	一目改						
		転換	基準	トレンド方向	価格位置	進行方向		転換	転換乖離	基準	基準乖離	トレンド	価格位置	進行
1	USDJPY 米ドル/円	111.497	112.801	下降	雲を下抜け	下降	111.140	111.497	-0.003	112.801	-0.015	DOWN	ABOVE CLOUD	DOWN
2	EURJPY ユーロ/円	120.420	120.555	新規下降(デッドクロス)	雲を下抜け	上昇	119.360	120.420	-0.009	120.555	-0.010	DOWN	ABOVE CLOUD	UP
3	SMNK 日経平均	19,239.80	19,300.05	下降	中立	下降	19,063.22	19,239.80	-0.01	19,300.05	-0.01	DOWN	ABOVE CLOUD	DOWN

TradeStationで作成。©TradeStation Technologies, Inc. All rights reserved/無断複写転載を禁じます。

ソースコピー時点で気付く方もいるかもしれませんが、コピー元ソースが英語版で日本語版じゃない感じです

そして雲の色やトレンドの出方が少し違います、基準線、転換線に違いはありません

一目均衡表・改を日本語版仕様に変更するのは手間がかかるので乖離率のみを表示して使用します

一目均衡表の見出し部分を右クリックでプロットを表示/非表示でいらない項目を非表示にします

銘柄コード	銘柄名	一目			一目改		現		
		トレンド方向	価格位置	進行方向	転換乖離	基準乖離			
1	USDJPY 米ドル/円	下降	雲を下抜け	下降	111.497	112.801	-0.003	-0.015	111.127
2	EURJPY ユーロ/円	新規下降(デッドクロス)	雲を下抜け	上昇	120.420	120.555	-0.009	-0.010	119.346
3	SMNK 日経平均	下降	中立	下降	19,239.80	19,300.05	-0.01	-0.01	19,063.22

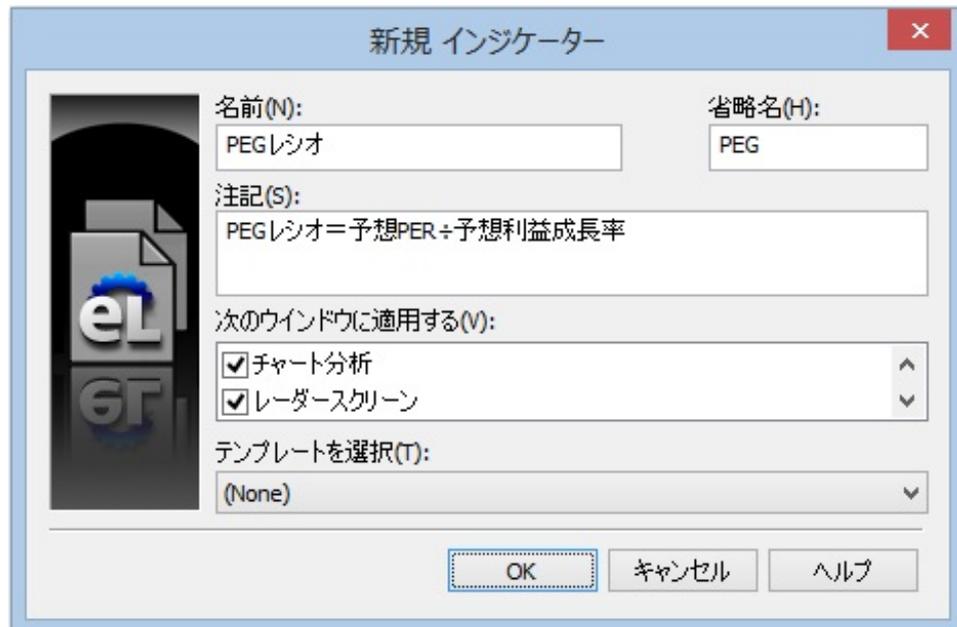
TradeStationで作成。©TradeStation Technologies, Inc. All rights reserved/無断複写転載を禁じます。

今まで現値（終値）と基準、転換を比較してたものが、乖離率で転換線・基準線割れがすぐに分かります

プログラミングに慣れたら基準、転換のGC、DCで色を変えたりともいいかもしれません

PEGレシオ

新しく任意のインジケータを作成します



● プログラム

```
Value1 = GetFundData("CE_PER", 0); // 予想PER : 現在の終値/一株当たり利益予想
Value2 = GetFundData("CE_OP_YOY", 0); // 予想営業利益成長率 : 営業利益_前期比
if Value2 <> 0 then Value3 = Value1 / Value2 ; // PEGレシオ = 予想PER ÷ 予想利益成長率
Plot1( Value1, "予想PER" );
Plot2( Value2, "営利率" );
Plot3( Value3, "PEG" );
```

● 解説

GetFundData (読取るデータ名, 過去のデータ期間数 (大きい程過去))

Value1 に最新の期間のファンダメンタルデータ「CE_PER」(予想PER)の数値を代入します

Value2 に最新の期間のファンダメンタルデータ「CE_OP_YOY」(予想営業利益前期比)の数値を成長率として代入します

もし Value2 がゼロじゃなかったら、Value1 ÷ Value2 の結果を Value3 に代入します

Plot1にValue1を、Plot2にValue2を、Plot3にValue3をそれぞれ代入して画面に表示します

"予想PER"、"営利率"、"PEG"は見出しとして表示されます

最小コードは6行です



PEGレシオは成長率も加味した指標で、

一般的に 1 以下が割安、2 以上が割高と言われていますので、

今回は数値によって見やすくするために、少し色を変えてみたいと思います

マイナス×マイナスでプラスになってしまったり、ゼロ以下のものも分かりやすく色を変化させます

色の指定は、色名でも色の番号でも出来ます 前面色が文字、背景色がセルみたいです

もし Value1 がゼロ以下だったら、Plot1をピンクで、それ以外は通常の色にします

Value2 もPlot2を同様の判断と色で処理

Value3 はゼロ以下だった場合はPlot3を赤色で、2 以上だった場合はオレンジ色で、

それ以外は通常の色にします

PEGレシオ (2)

- プログラム

```
// 色分け
```

```
if Value1 <= 0 then
```

```
    Plot1( Value1, "予想PER", MyColors("pink") )
```

```
else begin
```

```
    Plot1( Value1, "予想PER" );
```

```
End;
```

```
if Value2 <= 0 then
```

```
    Plot2( Value2, "営利率", MyColors("pink") )
```

```
else begin
```

```
    Plot2( Value2, "営利率" );
```

```
End;
```

```
Switch(Value3)
```

```
Begin
```

```
    Case <= 0:
```

```
        Plot3( Value3, "PEG", Red );
```

```
    Case >= 2:
```

```
        Plot3( Value3, "PEG", MyColors("orange") );
```

```
    Default:
```

```
        Plot3( Value3, "PEG" );
```

```
End;
```

3/31前場時点の2017年のIPOはこんな感じです、ユーザーローカルまだ寄りついてないですね(´д`)

銘柄 コード	銘柄名	PEGレシオ			株式(発行済株数・時価総額)	
		予想PER	営利率	PEG	発行済数(千)	時価総額(億)
1	3964 オークネット	14.60	0.1	208.57	26,269	339
2	3560 ほぼ日	30.37	0.2	151.85	2,250	99
3	3561 カの源ホールディングス	115.93	20.1	5.76	11,250	314
4	6694 ズーム	17.02	4.1	4.16	2,208	41
5	3562 No. 1	32.90	13.1	2.52	1,469	54
6	6545 インターネットインフィニティ	56.63	35.3	1.60	1,254	68
7	3479 ティーケーピー	44.93	29.8	1.51	4,730	597
8	3977 フュージョン	36.88	28.9	1.28	720	12
9	3978 マクロミル	19.02	19.6	0.97	38,622	696
10	3563 スシローグローバルホールディングス	15.91	18.8	0.85	27,459	950
11	1439 安江工務店	7.74	9.4	0.82	1,299	22
12	3976 シャノン	80.80	119.6	0.68	1,379	61
13	9519 レノバ	17.60	29.7	0.59	18,383	315
14	3979 うるる	91.44	169.4	0.54	3,050	120
15	3981 ビーグリー	20.10	42.8	0.47	5,879	142
16	9325 ファイズ	71.47	163.7	0.44	2,523	154
17	3557 ユナイテッド&コレクティブ	33.31	131.1	0.25	1,367	72
18	3558 ロコンド	34.38	192.8	0.18	5,170	111
19	3559 ピーバンドットコム	39.52	272.9	0.14	2,190	55
20	4597 ソレイジア・ファーマ	-11.83	-286.8	0.04	84,046	211
21	3984	0.00	15.9	0.00		
22	6175 ネットマーケティング	0.00	0.0	0.00	6,772	108
23	6546 フルテック	15.98	-28.0	-0.57	5,200	50
24	6544 ジャパンエレベーターサービスホールディングス	43.95	-31.1	-1.41	9,580	111
25	6543 日宣	19.78	-7.6	-2.62	1,938	48
26	6547 グリーンズ	11.73	-1.5	-7.87	12,000	174
27	3983 オロ	33.87	-0.6	-57.41	4,000	159
28						

グレアムのミックス係数

新しく任意のインジケータを作成します

新規 インジケータ

名前(N): グレアムのミックス係数 省略名(H): PEBR

注記(S): PER×PBR

次のウインドウに適用する(V):
 チャート分析
 レーダースクリーン

テンプレートを選択(T): (None)

OK キャンセル ヘルプ

● プログラム

```
Value1 = GetFundData("CE_PER", 0); // 予想PER : 現在の終値/一株当たり利益予想  
Value2 = GetFundData("CE_PBR", 0); // 予想PBR : PBR 現在の終値 / BPS 実績  
Value3 = Value1 * Value2; // PEGレシオ = 予想PER × 予想利PBR  
Plot1( Value1, "予想PER" );  
Plot2( Value2, "予想PBR" );  
Plot3( Value3, "MIX係数" );
```

● 解説

GetFundData (読取るデータ名, 過去のデータ期間数 (大きい程過去))

Value1 に最新の期間のファンダメンタルデータ「CE_PER」(予想PER)の数値を代入します

Value2 に最新の期間のファンダメンタルデータ「CE_PBR」(予想PBR)の数値を代入します

Value1 × Value2 の結果を Value3 に代入します

Plot1にValue1を、Plot2にValue2を、Plot3にValue3をそれぞれ代入して画面に表示します

"予想PER"、"予想PBR"、"MIX係数"は見出しとして表示されます

最小コードは5行です (Value3の行を削って、Value3の部分にValue1 * Value2を直接代入)



グレアムのミックス係数はみきまさんのブログにもあるように、

PER×PBRが22.5を超えてはいけないと言っていますので、

今回は数値によって見やすくするために、少し色を変えてみたいと思います

マイナス×マイナスでプラスになってしまったり、ゼロ以下のものも分かりやすく色を変化させます
色の指定は、色名でも色の番号でも出来ます 前面色が文字、背景色がセルみたいです

もし Value1、2 がゼロ以下だったら、Plot1、2をピンクで、それ以外は通常の色にします

Value3 はゼロ以下だった場合はPlot3を赤色で、22.5 以上だった場合はオレンジ色で、
半分の11.25 以上だった場合はマゼンダ色で、それ以外は通常の色にします

グレアムのミックス係数 (2)

銘柄コード	銘柄名	グレアムのミックス係数		
		予想PER	予想PBR	MIX係数 ▼
1	9325 ファイズ	71.47	40.20	2,873.09
2	3979 うるる	91.44	29.63	2,709.37
3	3561 カの源ホールディングス	115.93	12.20	1,414.35
4	6545 インターネットインフィニティ	56.63	20.10	1,138.26
5	3976 シャノン	80.80	9.44	762.75
6	3559 ピーバンドットコム	39.52	18.16	717.68
7	3479 ティーケーピー	44.93	13.48	605.66
8	3557 ユナイテッド&コレクティブ	33.31	13.26	441.69
9	6544 ジャパンエレベーターサービスホールディングス	43.95	10.03	440.82
10	3558 ロゴンド	34.38	7.16	246.16
11	3983 オロ	33.87	6.04	204.57
12	3562 No. 1	32.90	5.55	182.60
13	3977 フュージョン	36.88	4.01	147.89
14	3560 ほぼ日	30.37	4.41	133.93
15	9519 レノバ	17.60	4.75	83.60
16	3981 ビーグリー	20.10	3.78	75.98
17	3978 マクロミル	19.02	3.70	70.37
18	3563 スシローグローバルホールディングス	15.91	3.55	56.48
19	6543 日宣	19.78	2.54	50.24
20	6547 グリーنز	11.73	3.24	38.01
21	3964 オークネット	14.60	2.57	37.52
22	6546 フルテック	15.98	1.06	16.94
23	6694 ズーム	17.02	0.91	15.49
24	1439 安江工務店	7.74	1.77	13.70
25	3984 ユーザーローカル	0.00	0.00	0.00
26	6175 ネットマーケティング	0.00	0.00	0.00
27	4597 ソレイジア・ファーマ	-11.83	4.76	-56.31

TradeStationで作成。©TradeStation Technologies, Inc. All rights reserved/無断複写転載を禁じます。

● プログラム

// 色分け

if Value1 <= 0 then

Plot1(Value1, "予想PER", MyColors("pink"))

else begin

Plot1(Value1, "予想PER");

End;

if Value2 <= 0 then

Plot2(Value2, "予想PBR", MyColors("pink"))

else begin

Plot2(Value2, "予想PBR");

End;

Switch(Value3)

Begin

Case <= 0:

Plot3(Value3, "MIX係数", Red);

Case >= 22.5:

Plot3(Value3, "MIX係数", MyColors("orange"));

Case >= 11.25:

Plot3(Value3, "MIX係数", MyColors("Magenta"));

Default:

Plot3(Value3, "MIX係数");

End;

エクセル読み書き

浮動株、有利子負債、優待のデータが欲しくて、取り敢えずエクセルから読取りRSに表示させようと試みました
新しく任意のインジケータを作成します（レーダースクリーン用なのでチャート分析のチェックは外して下さい）
事前の作業として、RSに表示されている銘柄と同じ銘柄を同じ順序でエクセルに用意しておきます（コピペでOK）

- プログラム

```
Value1 = GetAppInfo(AiRow) ; // レーダースクリーンの行を取得
plot1(RSDData.sheets[1].CellsAsDouble[3,(Value1)+1], !("有利子負債(億)"));
// エクセルワークブック、シート、3列目、レーダースクリーンと同じ行を読み表示、(+1は見出し分)
Plot2(RSDData.sheets[1].CellsAsDouble[4,(Value1)+1], !("外国人保有%")); // 同じく4列目
Plot3(RSDData.sheets[1].CellsAsString[5,(Value1)+1], !("優待内容")); // 同じく5列目
```

- 解説

Value1にレーダースクリーンの行を取得し代入します
plot1にエクセルワークブックRSDDataのシート1番目、3列目の2行目のセルを読み代入します
plot2に同様に4列目の2行目のセルを読み代入します
plot3に同様に5列目の2行目のセルを読み代入します
最小コードは、読み書きする項目数+1です（行の取得用）

- 詳細

GetAppInfoは、指定キーワードに基づいて、呼び出しているアプリケーションの属性を指す数値を返す予約語です
キーワードにAiRowを指定するとレーダースクリーンの行番号を識別します
取得するデータ値によって CellsAs〇〇 の部分をデータのタイプと合わせる必要があります
文字値はCellsAsString、整数値はCellsAsint、少数値はCellsAsDoubleになります
プログラムコードがこんなに少なくてエクセルと読み書き出来るなんて驚きですが、その為には少し作業が必要です

表示ツールバーツールボックスのOffice-WorkbookをWクリック
プログラムコードエディタの下にコンポーネントトレイが出来るので、
そこにあるコンポーネントをクリックしてプロパティをクリックすると
以下の画面が表示されるので、ここにエクセルのファイル名のある場所、ワークブック名、
読みだけか、書き込みもするか等を選択、記入します

The screenshot shows the software interface with the code editor and the Properties window. The code editor contains the following code:

```
1 Value1 = GetAppInfo(AiRow) ;
2 plot1(RSDData.sheets[1].CellsAsDouble[3,(Value1)+1], !("有利子負債(億)"));
3 Plot2(RSDData.sheets[1].CellsAsDouble[4,(Value1)+1], !("外国人保有%"));
4 Plot3(RSDData.sheets[1].CellsAsString[5,(Value1)+1], !("優待内容"));
```

The Properties window shows the following properties for the RSDData component:

Property	Value
General	
FileName	"c:/Users/トレステ用.xls"
Shared	True
Visible	True
Load	True
SaveOnClose	False
design	
Name	RSDData

The Properties window also shows the Name property with the following description:

Name
データタイプ: string
The name of the component.
例: myObj.Name="myTimer"

エクセル読み書き (2)

プロパティの詳細な内容です

プロパティ

名前	タイプ	説明
ActiveSheet	Integer	アクティブなワークシートのインデックス番号(デフォルトプロパティ)を取得または設定します。
FileName	string	コンピュータ上の Excel スプレッドシートのパスおよびファイル名。
Load	boolean	真の場合、スプレッドシートへの接続を読み込んで開きます。
SaveOnClose	boolean	真の場合、接続を閉じたときにスプレッドシートに加えた変更を保存します。
Shared	boolean	真の場合、複数の Workbook オブジェクト全体で 1 つの Excel スプレッドシートを共有します。共有されない場合は偽です。
SheetCount	Integer	ワークブック内のシート数を指定します。
Sheets[Index]	object	使用可能なシートのインデックスに基づき Sheet を取得します。1ベースです。
Sheets[sSheetName]	object	指定したシート名文字列に基づき Sheet を取得します。
Visible	boolean	真の場合は接続時にスプレッドシートを表示し、偽の場合は表示せずにスプレッドシートをアップデートします。

メソッド

継承階層

プロパティ欄にエラーがなければ「RSDData」コンポーネントが働いてくれて先程の4行コードで済みます

内容は、表示デザイナージェネレーションコードでソースを表示することが出来ます

```
var: elsystem.office.excel.Workbook RSDData(NULL);
```

```
{ This method gets called by EasyLanguage one time
  at the beginning to create and initialize the components }
```

```
method override void InitializeComponent()
```

```
begin
```

```
    RSDData = new elsystem.office.excel.Workbook;
```

```
    //-----
```

```
    //rsdata
```

```
    //-----
```

```
    RSDData.FileName = "c:/Users/トステ用.xls";
```

```
    RSDData.Shared = true;
```

```
    RSDData.Visible = true;
```

```
    RSDData.Load = true;
```

```
    RSDData.SaveOnClose = false;
```

```
    RSDData.Name = "RSDData";
```

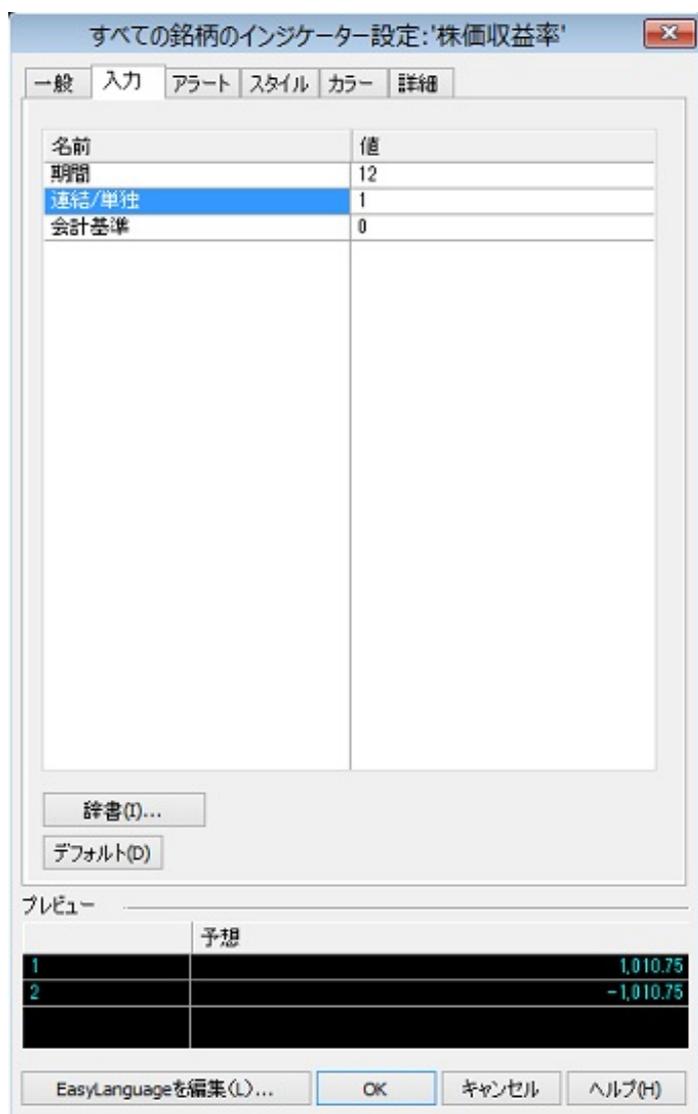
```
end;
```

このコードをプログラムにコピーしてコンポーネントを削除しても結果は同じになります

コード	略称	有利子負債(億)	外国人保有%	優待内容
1301	極洋	497.39	4.59	100株以上 5,000円相当の自社製品
1332	日本水産	2326.57	31.8	500株以上 3,000円相当自社製品
1333	マルハニチロ	2813.61	17.55	100株以上 自社グループ取扱商品
1376	カネコ種苗	9.35	5.41	
1377	サカタのタネ	50.32	11.18	100株以上 株主優待カタログ商品

銘柄	銘柄名	メモ	現	前比	前日比%	有利子負債(億)	外国人保有%	優待内容
1	1301	極洋	2,936	-6	-0.20%	497.39	4.59	100株以上 5,000円相当の自社製品
2	1332	日本水産	555	-11	-1.94%	2,326.57	31.80	500株以上 3,000円相当自社製品
3	1333	マルハニチロ	3,370	-65	-1.89%	2,813.61	17.55	100株以上 自社グループ取扱商品
4	1376	カネコ種苗	1,463	-30	-2.01%	9.35	5.41	
5	1377	サカタのタネ	3,485	-70	-1.97%	50.32	11.18	100株以上 株主優待カタログ商品

連結/単独問題



プログラムをいくつか作成して気づいたことですが、
今まで所々データ表示がないのは新しい会社か決算まだの会社かなと思って
たけど、入力値の連結か単独かでデータの場所が違うためとわかりました

殆どどちらかしかデータがなく、一括で変更するとどちらかが表示されなく
なる、ちょっとこれは問題です!(°_°)

本来連結/単独意識せず、
連結会社は連結データ、単独会社は単独データが出力されるもの

入力値の連結/単独を判断しているのは、米国版そのままなのか、設計ミス
かは分かりませんが、データのセットとプログラムとの整合性が取れていま
せん

その為ユーザーが一つ一つの企業ごと、指標ごとに連結か単独か判断して、
入力値を変更しなければいけません！

各プログラム内で整数値タイプで宣言して、初期値は 1 がセットされています

```
int ConsolidationLevel( 1 ) [DisplayName = "ConsolidationLevel", ToolTip =  
    "Enter 1 if consolidated data is desired; enter any other value for non-consolidated data." ],
```

各プログラム内では 1 が連結、それ以外は単独と判断してデータをセットしています

```
if ConsolidationLevel = 1 then  
    ConsLevel = ConsolidatedValue  
else  
    ConsLevel = NonConsolidatedValue;
```

公式にメールで問い合わせをしたところ、指標ごとに昇順か降順にまとめて選択すれば一括で変更出来るとの事でした
しかし、銘柄を殆ど入れ替えない人なら問題ないかもしれませんが、新しく企業を追加・変更する度にこの作業は煩わしすぎる(´д`;) ;
ただ、今のところは企業ごとに手入力に変更するしかないようです
実績データのでき方が今一つ把握出来ていないので、もう少し調べてから対応した指標を作成しないとイケないと思います

ファンダメンタルデータ

ヘルパー分析テクニックとストラテジーで検索-ファンダメンタルデータを実行すると以下の項目が表示されます

TS Platform Help - TradeStation

検索

ファンダメンタルデータ

分析テクニックとストラテジー

ファンダメンタルデータ - 日本株

次の表は、データタイプの直近のファンダメンタルデータのみ読み出すデータの名前、説明、タイプを示します。
ファンダメンタル分析の一環としてのファンダメンタルデータの使い方については、[「TradeStation」のファンダメンタルデータについて](#)を参照してください。
ファンダメンタルデータ名前は、その名前のパラメータのデータとステータスを取得するために、EasyLanguage 予約語とともに使用します。
説明列には、ファンダメンタルデータの簡単な説明が記載されます。
要求するデータのタイプをもとに EasyLanguage 予約語を使用する必要があります。

- [GetFundData](#) - Reads fundamental data of type Numeric.
- [GetFundDataAsString](#) - Reads fundamental data of type String.
- [GetFundDataAsBoolean](#) - Reads fundamental data of type Boolean.

フィールド名	説明	データタイプ	過去
CAB_ACCT_CLOSE_DATE	決算日	日付	Yes
CAB_ALLOTMENT_DATE	権利確定日	日付	Yes
CAB_CAP_AFTER_TRANSFERRERD	名義書換後資産	Double	Yes
CAB_DELIVERY_DATE	受渡および情報公開日	日付	Yes
CAB_EFFECTIVE_DATE	発効日および確定日	日付	Yes
CAB_END_DATE	名義書換および支払終了日	日付	Yes
CAB_EVENT_DATE	合併、株式交換、または分割日	日付	Yes
CAB_EX_RIGHT_DATE	権利落ち日	日付	Yes
CAB_FINAL_SECONDARY_DATE	再売出し最終日	日付	Yes
CAB_FORMER_PAR_VALUE	旧額面価格	Double	Yes
CAB_INITIAL_DATE	配当金計算の起算日	日付	Yes
CAB_ISSUE_DATE	新規株式発行日	日付	Yes
CAB_LAST_PRICE_WRIGHTS	最終権利付価格	Double	Yes
CAB_LIST_DATE	上場日および上場日変更基準	日付	Yes
CAB_LISTED_OUTSTANDING_SHARES	上場株式発行高、満期日前、ストックオプション行使による、総資本流出	Double	Yes
CAB_LISTED_SHARES_INCREASE	増加した上場株式、満期日後、優先株の転換、分配株	Double	Yes
CAB_LISTED_VS_OUTSTANDING_BASE	株式発行高数に対する上場株の基準線	Double	Yes
CAB_LISTED_VS_OUTSTANDING_RATIO	株式発行高数に対する上場株の比率	Double	Yes
CAB_NEW_PAR_VALUE	新規額面価格	Double	Yes

頭CRは実績データ、CEは予想データです (全ての項目にデータが入っているわけではありません確認必要)

現在分かっているのは予想データは簡単に取得出来ますが、実績データは日付を見て判断する必要があります

例：実績EPS、予想EPS

インジケーター、カスタムでCR_EPS、CE_EPSを入力した場合、プログラムでGetFundDataで取得する場合を比較確認

Value1 = GetFundData("CR_EPS", 0);

Value2 = GetFundData("CE_EPS", 0);

Plot1(Value1, !("CR_EPS"));

Plot2(Value2, !("CE_EPS"));

銘柄コード	銘柄名	一株益		カスタムファンダメンタル		ファンダメンタルデータ(日本株)	
		実績	予想	財務基礎データを選択して描画	カスタムファンダメンタル:2 財務基礎データを選択して描画	CR_EPS	CE_EPS
1 1439	東江工務店					214.8	172.5
2 3479	ディーケーピー	219.0	308.1	219.0	308.1	314.7	308.1
3 3557	ユナイテッド&コレクション					78.1	193.8
4 3558	ロコント					359.6	194.9
5 3559	ビーバンドットコム					53.0	66.7
6 3560	ほく日					176.6	155.5
7 3561	力の原ホールディングス	12.2	25.4	12.2	25.4	13.2	25.4
8 3562	No. 1	190.2	213.7	190.2	213.7	190.6	213.7
9 3563	スシローグローバルホールディングス	90.1	214.3	90.1	214.3	53.4	214.3
10 3964	オークネット	90.8	89.7	90.8	89.7	90.8	89.7
11 3976	シャノン	35.8	50.1	35.8	50.1	-17.3	50.1
12 3977	キュージョン					68.2	53.9
13 3978	マクミル	74.6	97.1	74.6	97.1	48.9	97.1
14 3979	うるる	-112.4	47.7	-112.4	47.7	58.8	47.7
15 3981	ビーグリー					74.1	115.9
16 3983	オロ	153.7	121.8	153.7	121.8	153.7	121.8
17 3984	ユーザーローカル					38.1	69.8
18 4597	ソレイジア・ファーマ	-18.5	-22.6	-18.5	-22.6	-18.5	-22.6
19 6175	ネットマーケティング	27.2	39.2	27.2	39.2	21.4	39.2
20 6543	日宣	117.6	141.5	117.6	141.5	153.4	141.5
21 6544	ジャパコエレベーターサービスホールディングス	50.6	31.9	50.6	31.9	18.8	31.9
22 6545	インターネットインフィニティ					63.9	98.1
23 6548	フルテック	106.1	71.6	106.1	71.6	39.2	71.6
24 6547	クリンズ	129.0	140.6	129.0	140.6	125.0	140.6
25 6694	ズーム	89.7	110.7	89.7	110.7	89.7	110.7
26 9325	フェイス					83.3	84.0
27 9519	レノバ	18.9	99.1	18.9	99.1	126.9	99.1
28							

予想のデータは合っていますが、実績のデータは微妙に違っているのが分かります

本決算と四半期の違いなのか、もう少しどのように実績データが作成されているのか調べてみたいと分かりません

ファンダメンタルデータ (2)

ヘルプオブジェクトリファレンスでファンダメンタルデータを検索するとまた違う項目が出てきます

ファンダメンタルデータ
検索

ランク	タイトル
1	例外メッセージ
2	FundamentalQuotesProvider
3	FundamentalUpdateReader

FundamentalFields クラス

基本クォートフィールド名のリストを含みます。このリストは、FundamentalQuotesまたは FundamentalQuotesProvider フィールドコレクションから Quote["Name"] 値の読み取るために使用できます。

通常、数値は DoubleValue として、Date および Time は DateValue として、名前説明は StringValue として参照されます。特定の Quote["Name"] 値を読み取るために必要なプロパティのタイプを識別するには、以下の **値タイプ** を参照してください。これらの値タイプは、特定のフィールド名に関する EL 辞書説明ページの「例」セクションにもあります。

基本クォートの中には前のレポート期間の過去データを含むものもあるため、'typeValue' プロパティには遡る期間の数を参照するインデックスを含める必要があります。ここで、[0] は現在の期間の最後のクォート、[1] は 1 つ前の期間を意味します。また、'typeValueLast' プロパティで現在の期間の最後のクォートを読み取ることもできます。

```
Value1 = FundamentalQuotesProvider1.Quote["ATOT"].DoubleValue[1]; // gets the Total Assets from 1 report period ago
Value2 = FundamentalQuotesProvider1.Quote["AACR"].DoubleValue[0]; // gets the current period Accounts Receivable Net Trade
Plot1(FundamentalQuotesProvider1.Quote["YRAGODATE1"].DateValueLast) // plots the date of the 1 Year Ago Price
Plot2(FundamentalQuotesProvider1.Quote["AskExchange"].StringValueLast) // plots the exchange name of the last ask (offer)
```

☑ すべてのフィールドの基本データを特定のシンボルに使用できるとは限りません。値を計算またはプロットする前に、HasQuoteData("ATOT") メソッドを使用して、現在のプロバイダオブジェクトにより特定のクォートにデータを利用できるかどうかを判断することをお勧めします。

名前空間: tsdata.marketdata

[\[すべて折りたたむ\]](#)

プロパティ

列をソートするには、列タイトルをクリックして昇順または降順でアイテムを表示します。

名前	説明	値タイプ	履歴
A1FCF	フリーキャッシュフロー - 最も古い会計年度	double	
A1FCFSHR	一株当たりフリーキャッシュフロー - 直近会計年度	double	
A2FCFSHR	一株当たりフリーキャッシュフロー - 直近会計年度 - 1	double	
A2NETMRGN	純利益率 (%) - 2 番目に古い会計年度	double	
AACR	売掛金 - 純取引	double	あり
AAGA	累積のれん償却費	double	あり
AAII	未収投資収益	double	あり
AAMT	累積無形資産償却	double	あり
AARG	売掛金 - 総取引	double	あり
AASSTURN	資産回転率 - 直近会計年度	double	
ABEPSXCLXO	特別損益項目を除いた基本 EPS - 直近会計年度	double	
ABVPS	一株当たり純資産 (株主資本) - 直近会計年度	double	
ACAC	得意先引受	double	あり
ACAE	現金および現金同等物	double	あり
ACAPSPPS	一株当たり資本支出 (直近会計年度)	double	

浮動株の項目とか使用したいのですが、米国版らしくHELPのやり方やGetFundDataではエラーになり取得できません

```
Value1 = FundamentalQuotesP1.Quote["FLOAT"].DoubleValue[0];
```

```
Plot1( Value1, !( "FLOAT" ) );
```

```

E エラー:
E elsystem.InvalidOperationException: [プロバイダー]のデータは、オブジェクトが[ロード済み]になるまでアクセスできません
E コールスタック:
E ファンダメンタルデータ(日本株) main()
E ELRte
E
E
E
```

現状は使用できない感じです(´д`)

連結/単独問題にも絡んで、実績データの違いや出来方等、把握出来たらまた更新したいと思えます

少しまとまった時間がないとやる気になりませんが(´д`;))

ファンダメンタルデータ (3)

営業利益・進捗インジケータを作る際に、実績データを検証して前回謎だった部分が判明したのでまとめます

銘柄コード	銘柄名	宮利		宮利益・進捗					直近決算日	決算日	カスタム... データを選択して進	ファンダ確認							
		実績	予想	1Q	2Q	3Q	4Q	進捗%				FundValue	Count	PeriodsAgo Value	Post Date	期間	会計基準	連結/単独	TermType
1	3289-TS 東急不動産ホールディングス	68,750	73,000	9,356	15,988	16,786	73,000	57.7%	2017/02/09	3/31	68,750	68,750	15	16,786	2017/02/09	3	JSTD	Consolidated	QuarterTerm
2	3479-TS ティーケービー	2,694	3,271				3,271		2017/04/13	2/28	2,694	2,694	4	2,830	2017/04/13	12	JSTD	NonConsolidated	FullTerm
3	3559-TS ビーバンドットコム	59	220			163	220	74.1%	2017/03/09	3/31			2	163	2017/03/09	9	JSTD	NonConsolidated	QuarterTerm
4	3560-JQ 日債日	499	500	541	178		500	143.8%	2017/04/13	8/31			3	719	2017/04/13	6	JSTD	NonConsolidated	MidTerm
5	3561-TS カの源ホールディングス	502	603			493	603	81.8%	2017/03/21	3/31	502	502	2	493	2017/03/21	9	JSTD	Consolidated	QuarterTerm
6	3562-JQ No. 1	304	330				330		2017/04/12	2/28	304	304	4	275	2017/04/12	12	JSTD	NonConsolidated	FullTerm

TradeStationで作成。©TradeStation Technologies, Inc. All rights reserved.無断転写複製を禁じます

ファンダメンタルデータは主に、カウント、シンボル（銘柄コードと市場）、フィールド（項目）、値、日付で1件のデータになっています

他にヘルプにも載っていない項目で確認できているのが、期間、会計基準、連結/単独、期間タイプがあります

既存のインジケータでよくある上記入力項目3つは、この非表示の項目とマッチングしてプロットされています

東急不動産を例に営業利益の実績データを見てみると、

GetFundData("CR_OP", 0)やFundamentalQuotesProvider等で最新のデータを取得する場合

シンボル3289-TSと項目CR_OPとカウント0でデータを取得する為、最新実績データの3Qの値16,786を取得します

既存の営業利益インジケータの場合、シンボルと項目は同じですが、入力項目で選択しているため

(デフォルトでは期間12、連結1、会計基準0) 本決算の値68,750が表示されるようになっていきます

ファンダメンタルデータ 日本株								
銘柄	シンボル	3289-TS		東急不動産ホールディングス				
項目	フィールド名	CR_OP		営業利益実績データ				
入力項目								
件数	遡り期間に該当する値	項目	銘柄	発表日	期間	会計基準	連結/単独	期間タイプ
Count総数	Data	Fields	Symbol	PostDate	Month	Acct	Consol	TermType
15					非表示項目			
Count明細	PeriodsAgoValue			PostDate	Month	Acct	Consol	TermType
0	16,786	CR_OP	3289-TS	2017/2/9	3	JSTD	Consolidated	QuarterTerm
1	25,344	CR_OP	3289-TS	2016/11/9	6	JSTD	Consolidated	MidTerm
2	9,356	CR_OP	3289-TS	2016/8/1	3	JSTD	Consolidated	QuarterTerm
3	11,889	CR_OP	3289-TS	2016/5/11	12	JSTD	NonConsolidated	FullTerm
4	68,750	CR_OP	3289-TS	2016/5/11	12	JSTD	Consolidated	FullTerm
5	17,715	CR_OP	3289-TS	2016/2/8	3	JSTD	Consolidated	QuarterTerm
6	29,547	CR_OP	3289-TS	2015/11/9	6	JSTD	Consolidated	MidTerm
7	11,505	CR_OP	3289-TS	2015/7/31	3	JSTD	Consolidated	QuarterTerm
8	10,096	CR_OP	3289-TS	2015/5/12	12	JSTD	NonConsolidated	FullTerm
9	63,300	CR_OP	3289-TS	2015/5/12	12	JSTD	Consolidated	FullTerm
10	12,913	CR_OP	3289-TS	2015/2/6	3	JSTD	Consolidated	QuarterTerm
11	24,940	CR_OP	3289-TS	2014/11/7	6	JSTD	Consolidated	MidTerm
12	9,161	CR_OP	3289-TS	2014/7/31	3	JSTD	Consolidated	QuarterTerm
13	61,433	CR_OP	3289-TS	2014/5/12	12	JSTD	Consolidated	FullTerm
14	10,018	CR_OP	3289-TS	2014/1/31	3	JSTD	Consolidated	QuarterTerm

既存インジケータのカスタムファンダなどで入力する場合も同じで、期間12、連結/単独1、会計基準0、遡り0で入力すると

カウント明細4のデータが最新の決算連結データとして表示されるようにプログラムで制御されていて

違う会計基準や期間、非連結のデータは省かれて、遡り1はカウント明細9のデータが表示されるようになっていきます

上記の例で、期間3、連結/単独1、会計基準0、遡り1で1Qの9,356が、遡り0で3Qの16,786が表示されます

これによりGetFundData等で取得する場合と差異が生じているようなので、使い分けやプログラムでの制御が必要ですよ

本決算の後に非連結のデータも同日で作成されている場合があります

期間の数字はその期間でまとめられている的な意味だと思うので各Quarterごと3か月分のデータではなく

基本的に1Q、3Qは3か月でQuarter、2Qは1Q+2Q分の6か月MidTermで中間決算、4Qは12か月FullTermで本決算です

再上場や新規IPO銘柄や決算期の変更等で3、6、12以外の月数でまとめられているデータもたまに存在しています

実績は4半期ごと作成されていて、予想は本決算時に殆どが連結は連結、単独は単独の次期予想としてデータが作成されると思う

上方や下方があった場合もそれが最新になるので、予想の方はそれほどGetFundData等で取得する場合と差異がないと思います

たまに連結なのに単独の中間期予想のデータがある会社もありました(汗)

今回の調査で連結/単独の部分や四半期や中間、本決算等のデータ部分がある程度理解できたのでよかったです

執筆順番は逆ですが、こうして営業利益・進捗インジケータの作成に取り掛かる事になります

投資の女神様

相場の女神さまから投資の女神さまへ、4月からリニューアルされた向後はるみさんにフォローされたので、女神さま投資法はありなのか、パフォーマンスはどうか気になり簡単なインジケータを作成してみました
 入力した株価を基準に、差額と%を表示します、売買ったつもりノーポジシミュレーションとしても使えます
 株価をマイナス入力した場合、空売りした事にしてプラスマイナス逆に表示するようにしました
 ご本人が聞いてくれた方だけという事でブログにもあまりラジオで言った個別銘柄を表示していませんので銘柄は隠させていただきます
 パフォーマンスは翌日寄付の値から計っているそうです

4/12の1銘柄の-47.2%は分割によるもので分割対応はしていませんので、基準株価を自分で入力し直す必要があります

銘柄コード	銘柄名	投資の女神				現在値	前日比	前日比%	52週高値安値ライン		
		メモ	株価	差額	差%				52週高値	52週安値	現在の%位置
03/01			690	215	31.2%	905	-32	-3.42%	1,055	406	76.89%
03/01			817	248	30.4%	1,065	-192	-15.27%	1,600	474	52.49%
03/08			958	-182	-19.0%	776	-5	-0.64%	1,384	605	21.95%
03/08			892	-30	-3.4%	862	42	5.12%	965	290	84.74%
03/15			2,349	76	3.2%	2,425	-73	-2.92%	2,838	1,110	76.10%
03/15			3,820	50	1.3%	3,870	55	1.44%	4,690	2,550	61.68%
03/29			1,399	-191	-13.7%	1,208	-19	-1.55%	1,415	884	61.02%
03/29			3,750	130	3.5%	3,880	270	7.48%	4,375	1,155	84.63%
04/05			1,270	82	6.5%	1,352	-22	-1.60%	1,582	575	77.16%
04/05			3,415	525	15.4%	3,940	70	1.81%	4,265	2,950	75.29%
04/12			670	120	17.9%	790	-47	-5.62%	843	350	89.25%
04/12			3,695	-1,745	-47.2%	1,950	-25	-1.27%	2,140	640	87.33%
04/19			1,690	34	2.0%	1,724	-54	-3.04%	1,877	916	84.08%
04/19			989	33	3.3%	1,022	9	0.89%	1,035	531	97.42%
04/26			7,200	-540	-7.5%	6,660	0	0.00%	7,500	5,630	55.08%
04/26			904	23	2.5%	927	-1	-0.11%	1,396	267	58.46%
04/26			5,190	-320	-6.2%	4,870	-200	-3.94%	5,580	3,500	65.87%
04/26			-5,190	320	6.2%	4,870	-200	-3.94%	5,580	3,500	65.87%

TradeStationで作成。©TradeStation Technologies, Inc. All rights reserved/無断複写転載を禁じます。

inputs:

```
string Memo("") [DisplayName = "メモ", ToolTip = "日付・株数等備考"],
int Price(0) [DisplayName = "基準株価", ToolTip = "評価基準株価"],
UpColor( UpColorDefault ) [DisplayName = "上昇色", ToolTip = "上昇時"],
UpColor( DownColorDefault ) [DisplayName = "下降色", ToolTip = "下降時"];
```

```
Plot1( Memo, !( "メモ" ) );
Plot2( Price, !( "株価" ) );
//空売り対応（株価をマイナス入力で空売りした事にする）
If Price < 0 then begin
    Value1 = (( Last + Price ) / Price );
    Plot3((( Last + Price ) * (-1), !( "差額" ) );
end else if Price > 0 then begin
    Value1 = (( Last - Price ) / Price );
    Plot3(( Last - Price ), !( "差額" ) );
end;
Plot4( Value1, !( "差%" ) );

//上下色分け
if Value1 > 0 then begin
    SetPlotColor( 3, UpColor );
    SetPlotColor( 4, UpColor );
end else if Value1 < 0 then begin
    SetPlotColor( 3, DnColor );
    SetPlotColor( 4, DnColor );
end;
```

営業利益・進捗

実績データの検証をしてみて通期のデータだけでなく、四半期データも存在する事が分かったので、
 少数の銘柄を手入力で管理していた営業利益の進捗インジケータをそのデータを読み込んで作成してみました
 既存の営業利益インジケータは実績は前期、予想は今期で、今期の実績がないので隙間を埋めるインジケータになると思います
 無駄にサブルーチンとか使ってみたりして過去最長100ステップ超えとなっております(°_°)安心して下さいコピペできますよ！

	銘柄コード	銘柄名	宮利		宮利益・進捗					決算日	
			実績	予想	1Q	2Q	3Q	4Q	進捗%		直近決算日
24	6545-TS	インターネットインフィニティ	119	161			105	161	65.2%	2017/03/21	3/31
25	6546-TS	フルテック	806	580			416	580	71.7%	2017/03/22	3/31
26	6547-TS	グリーンズ	2,278	2,244		1,789	-76	2,244	76.3%	2017/04/28	6/30
27	6548-TS	旅工房	230	253			328	253	129.6%	2017/04/18	3/31
28	6694-JQ	ズーム	220	229				229		2017/03/28	12/31
29	6758-TS	ソニー	288,702	500,000				500,000		2017/04/28	3/31
30	7940-TS	ウェブロックホールディング	740	1,200			1,013	1,200	84.4%	2017/04/10	3/31
31	9325-TS	フェイス	113	298			293	298	98.3%	2017/03/15	3/31
32	9983-TS	ファーストリテイリング	127,292	175,000	88,591	42,066		175,000	74.7%	2017/04/13	8/31
33	日経平均株価 (225)										
34	1332-TS	日本水産	19,442	21,500	4,174	5,638	8,902	21,500	87.0%	2017/02/03	3/31
35	1333-TS	マルハニチロ	16,972	26,000	7,239	7,559	10,676	26,000	98.0%	2017/02/06	3/31
36	1605-TS	国際石油開発帝石	390,139	335,000	70,773	67,299	97,736	335,000	70.4%	2017/02/10	3/31
37	1721-TS	コムシスホールディングス	23,849	25,000	1,330	4,971	4,706	25,000	44.0%	2017/02/07	3/31
38	1801-TS	大成建設	117,468	140,800	16,064	36,450	51,882	140,800	74.1%	2017/02/08	3/31
39	1802-TS	大林組	106,380	132,000	22,986	36,162	37,984	132,000	73.6%	2017/02/14	3/31
40	1803-TS	清水建設	94,668	128,500	21,989	29,748	36,314	128,500	68.5%	2017/02/14	3/31
41	1808-TS	長谷工コーポレーション	68,762	86,000	14,592	21,425	22,808	86,000	68.4%	2017/02/10	3/31

TradeStationで作成。©TradeStation Technologies, Inc. All rights reserved./無断複写転載を禁じます。

コードが長いので先にザックリ概要を説明すると

最初にクラスの参照、変数や定数の宣言をして、サブルーチンの記述、

その後に銘柄コード毎に1回だけの処理の中でサブルーチンの処理をしてデータをプロットしています

入力項目は"各期合計"、"予想営業利益"両方ともデフォルト0でセット

"各期合計"、0で各期毎のデータを表示し、0以外で合計表示（2Q、3Qは前のQまでの値を含んで表示）

"予想営業利益"、0の時は最新の予想営業利益を表示、連結/単独の関係で違う数字やゼロ、数字を変えたい時等は入力した数字を表示

4Qの所に予想営業利益を表示しています

進捗の%は各期×25%を超えていれば上昇色、超えてなければ下降色にしています（色は入力項目で変更可）

● プログラム

```
using tsdata.marketdata;
```

```
using elsystem;
```

```
inputs: //入力変数宣言
```

```
int QTotal(0) [DisplayName = "各期合計", ToolTip = "0=各期毎表示,0≠各期合計表示"],
```

```
int Ex_Profit(0) [DisplayName = "予想営業利益", ToolTip = "ゼロや稀に違う予想営業利益対応"],
```

```
UpColor( UpColorDefault ) [DisplayName = "上昇色", ToolTip = "上昇時"],
```

```
DnColor( DownColorDefault ) [DisplayName = "下降色", ToolTip = "下降時"];
```

営業利益・進捗（2）

```
Vars: FundamentalQuotesProvider FQP( null ), //変数宣言
double Quarter1( 0 ), //1Q実績
double Quarter2( 0 ), //2Q実績
double Quarter3( 0 ), //3Q実績
double Quarter4( 0 ), //4Q実績
DateTime Kessan( null ); //直近決算日

constants: //定数宣言
string KeyTermType( "TermType" ),
string KeyFullTerm( "FullTerm" ),
string KeyMonthsReported( "MonthsReported" ),
string KeyActualFieldName( "CR_Op" );

method bool GetQuoteVal { GetQuoteVal サブルーチン }
( FundamentalQuote fq,DateTime tempdt,out double Quarter1,out double Quarter2,out double Quarter3,out double Quarter4,out DateTime Kessan )
variables: //サブルーチン内変数宣言
int fqCount, //カウント
int fqMonth, //決算期間
bool QuoteFound, //データの存在有無
DateTime LastDateTime, //直近本決算日
DateTime fqDateTime; //発表日
begin
if fq = null or tempdt = null then return false; //データ存在無し、falseでサブルーチン終了
QuoteFound = false; //データの存在無セット
LastDateTime = DateTime.Create( 1900, 1, 1 ); //直近本決算日セット1900/01/01
for fqCount = fq.Count - 1 downto 0 begin //カウント分繰り返し古いデータから逆順に確認
fqDateTime = fq.PostDate[fqCount]; //発表日セット
Kessan = fq.PostDate[fqCount]; //直近決算日セット
fqMonth = fq.ExtendedProperties[fqCount][KeyMonthsReported].IntegerValue; //期間セット
if fq.ExtendedProperties[fqCount][KeyTermType].StringValue = KeyFullTerm then begin //FullTerm 本決算判断
if tempdt >= fqDateTime and fqDateTime >= LastDateTime then begin //現在日時以下、直近本決算日以上
LastDateTime = fqDateTime; //直近本決算日セット
Quarter1 = 0; //各Q実績リセット
Quarter2 = 0;
Quarter3 = 0;
end;
end else if tempdt >= fqDateTime and fqDateTime >= LastDateTime then begin //FullTerm本決算以外
switch(fqMonth) begin //各Q実績セット
case = 3:
if Quarter2 = 0 then begin Quarter1 = fq.DoubleValue[fqCount]; //期間3で2Qが0の時、1Qにセット
end else if Quarter2 <> 0 then Quarter3 = fq.DoubleValue[fqCount]; //期間3で2Qが0以外の時、3Qにセット
case = 6:
Quarter2 = fq.DoubleValue[fqCount]; //期間6の時、2Qにセット
default:
Quarter3 = fq.DoubleValue[fqCount]; //期間3、6以外の時、3Qにセット
end;{ switch }
end;
QuoteFound = true; //データの存在有セット
end;{ for downto }
return QuoteFound; //データ存在有りはtrue、無しはfalseでサブルーチン終了
end;{ GetQuoteVal method }
```

営業利益・進捗（3）

```
once begin
  FQP = new FundamentalQuotesProvider();           //クラスのオブジェクトを作成し、そのクラスのコンストラクターを起動
  FQP.Symbol = Symbol;                             //銘柄指定
  FQP.Fields += KeyActualFieldName;                //項目指定 CR_Op 実績営業利益
  FQP.LoadProvider();                              //データプロバイダーへの接続を確立
end;{ once }
//GetQuoteVal サブルーチン実行、データ存在時のみ以下実行
if GetQuoteVal( FQP[KeyActualFieldName], BarDateTime, Quarter1, Quarter2, Quarter3, Quarter4, Kessan ) then begin
  if Quarter1 <> 0 then Plot1( Quarter1, !( "1Q" ) );
  if QTotal = 0 then begin                          //各期毎表示の場合
    if Quarter2 <> 0 then Plot2( Quarter2 - Quarter1, !( "2Q" ) );
    if Quarter3 <> 0 then Plot3( Quarter3, !( "3Q" ) );
  end else if QTotal <> 0 then begin                  //各期合計表示の場合
    if Quarter2 <> 0 then Plot2( Quarter2, !( "2Q" ) );
    if Quarter3 <> 0 then Plot3( Quarter3 + Quarter2, !( "3Q" ) );
  end;
  if Ex_Profit = 0 then begin                        //入力予想営業利益ゼロの時
    Quarter4 = GetFundData("CE_OP", 0);             //最新予想営業利益
  end else if Ex_Profit <> 0 then begin               //入力予想営業利益ゼロ以外の時
    Quarter4 = Ex_Profit;                            //入力した予想営業利益
  end;
  if Quarter4 <> 0 then Plot4( Quarter4, !( "4Q" ) );
  if Kessan <> 0 then Plot6( Kessan.ToString(), !( "直近決算日" ) );
//進捗%
  if Plot1 <> 0 and Quarter4 <> 0 then begin
    Value1 = ( Plot1 / Quarter4 );
    Value2 = 0.25;
  end;
  if Plot2 <> 0 and Quarter4 <> 0 then begin
    if QTotal = 0 then begin Value1 = ( Quarter2 / Quarter4 ); //各期毎表示の場合
    end else if QTotal <> 0 then Value1 = ( Plot2 / Quarter4 ); //各期合計表示の場合
    Value2 = 0.5;
  end;
  if Plot3 <> 0 and Quarter4 <> 0 then begin
    if QTotal = 0 then begin Value1 = (( Quarter3 + Quarter2 ) / Quarter4 ); //各期毎表示の場合
    end else if QTotal <> 0 then Value1 = ( Plot3 / Quarter4 ); //各期合計表示の場合
    Value2 = 0.75;
  end;
  if Value1 <> 0 then Plot5( Value1, !( "進捗%" ) );
//上下色分け
  if Value1 > Value2 then begin SetPlotColor( 5, UpColor ); //上昇色：各期×25%より大きい場合
  end else if Value1 < Value2 and Value1 > 0 then begin SetPlotColor( 5, DnColor ); //下降色：各期×25%より小さい場合
  end;
end;{ GetQuoteVal }
```

ボラティリティ

先物用にとっってボラのインジケータを調べてみたら%は見つかったけど、値幅自体を知るのが見つからなかったので自作しました

銘柄コード	銘柄名	足種	現	前比	前日比%	始	高	安	ボラ								
									始値	高値	安値	中値	高値始値	始値安値	今	短期平均	長期平均
1 7203	トヨタ自動車	15分	5,893.0	-17.0	-0.29%	5,896.0	5,906.0	5,867.0	5,885	5,893	5,885	5,889	8	0	8	10	9
2 7203	トヨタ自動車	60分	5,893.0	-17.0	-0.29%	5,896.0	5,906.0	5,867.0	5,885	5,893	5,885	5,889	8	0	8	15	15
3 7203	トヨタ自動車	240分	5,893.0	-17.0	-0.29%	5,896.0	5,906.0	5,867.0	5,875	5,893	5,867	5,880	18	8	26	29	33
4 7203	トヨタ自動車	日	5,893.0	-17.0	-0.29%	5,896.0	5,906.0	5,867.0	5,896	5,906	5,867	5,887	10	29	39	39	52
5 7203	トヨタ自動車	週	5,893.0	-17.0	-0.29%	5,896.0	5,906.0	5,867.0	5,850	5,934	5,838	5,886	84	12	96	144	190
6 7203	トヨタ自動車	月	5,893.0	-17.0	-0.29%	5,896.0	5,906.0	5,867.0	5,935	6,097	5,768	5,933	162	167	329	396	672

既存の始値、高値、安値も分足や週足など違う時間軸でも日足ベースの値が表示されるので追加しました

入力項目は短期と長期の期間で平均値を算出、色は中値が現在値と、今は短期と、短期は長期と比較して分けています

- プログラム

input:

```
Length2( 5 ) [DisplayName = "短期数"],
Length3( 20 ) [DisplayName = "長期数"];
```

variables:

```
Counter( 0 ),
Volty( 0 ),
Volty1( 0 ),
Volty2( 0 ),
Volty3( 0 );
```

//ボラ計算

```
Volty1 = High[0] - Low[0];
```

```
Volty = 0;
```

```
for Counter = 0 to Length2 - 1 begin
```

```
    Volty = Volty + ( High[Counter] - Low[Counter] );
```

```
end;
```

```
if Length2 <> 0 then Volty2 = Volty / Length2;
```

```
Volty = 0;
```

```
for Counter = 0 to Length3 - 1 begin
```

```
    Volty = Volty + ( High[Counter] - Low[Counter] );
```

```
end;
```

```
if Length3 <> 0 then Volty3 = Volty / Length3;
```

//表示

```
Plot1( Open[0], !( "始値" ) );
```

```
Plot2( High[0], !( "高値" ) );
```

```
Plot3( Low[0], !( "安値" ) );
```

```
Plot4( Low + ( Volty1 / 2 ), !( "中値" ) );
```

```
Plot5( High - Open, !( "高値始値" ) );
```

```
Plot6( Open - Low, !( "始値安値" ) );
```

```
Plot7( Volty1, !( "今" ) );
```

```
Plot8( Volty2, !( "短期平均" ) );
```

```
Plot9( Volty3, !( "長期平均" ) );
```

//色分け

```
if Plot4 > Last then begin SetPlotColor( 4, UpColorDefault );
```

```
end else if Plot4 < Last then begin SetPlotColor( 4, DownColorDefault ); end;
```

```
if Plot7 > Plot8 then begin SetPlotColor( 7, UpColorDefault );
```

```
end else if Plot7 < Plot8 then begin SetPlotColor( 7, DownColorDefault ); end;
```

```
if Plot8 > Plot9 then begin SetPlotColor( 8, UpColorDefault );
```

```
end else if Plot8 < Plot9 then begin SetPlotColor( 8, DownColorDefault ); end;
```

基準日変動率%

日経のMIDDLE200が公開されてからどの位変動してるのを知りたかったのでキリよく3/31の終値からの変動率を知る為に作成しました
 既存のインジケータは過去何本前の終値が調べて入力し変動率を表示できますが日が経つにつれて基準日も動いてしまいます
 何本前か調べるのも、1日毎に+1するのも面倒なので基準日入力して、その終値から計算できないものかと思いましたが、
 基準日の終値を取得する方法が分からず、出来たばかりのフォーラムでココアさんに回答頂き解決、ありがとうございました！

- ※ 17/8/15 終値と終値で比較していたものを、基準値と比較値を入力項目に追加、始値と終値、高値と安値、等比較出来るように変更
- ※ 現在値 Last も入力できますが、過去の日付で入力しても今日の現在値を持って来るため基準値としては適当ではありません
- ※ 出来高 Volume は当然ながら、出来高同士で比較しないとおかしな比較数値になります(°ω°)
- ※ 基準値は今日を含む過去の指定した基準日の値、比較値は最新の値なので今日の値になります (画像は旧バージョン)

	銘柄コード	銘柄名	現	前比	前日比%	基準日変動率%			
						基準日	基準日終値	差額	変動率%
1	3810	サイバーステップ	6,350	-40	-0.63%	2017/03/31	997	5,353	536.91%
2	3825	リミックスポイント	1,540	-49	-3.08%	2017/03/31	284	1,256	442.25%
3	4563	アンジェス	780	3	0.39%	2017/03/31	251	529	210.76%
4	3346	21LADY	315	-19	-5.69%	2017/03/31	104	211	202.88%
5	4288	アズジェント	4,135	35	0.85%	2017/03/31	1,564	2,571	164.39%
6	2928	RIZAPグループ	1,961	12	0.62%	2017/03/31	844	1,117	132.35%
7	4579	ラクオリア創薬	1,036	-10	-0.96%	2017/03/31	450	586	130.22%
8	6628	オンキヨー	258	17	7.05%	2017/03/31	129	129	100.00%
9	3634	ソケッツ	2,339	154	7.05%	2017/03/31	1,234	1,105	89.55%
10	6239	ナガオカ	1,227	20	1.66%	2017/03/31	650	577	88.77%

TradeStationで作成。©TradeStation Technologies, Inc. All rights reserved/無断複写転載を禁じます。

● プログラム

```
inputs:
    //入力変数宣言
    int Hizuke( 1170331 ) [DisplayName = "過去日 (EL形式 YYYY MM DD) "],
    double RefVal( Close ) [DisplayName = "基準値 (Open,Close,High,Low,Volume等) "],
    double ComVal( Close ) [DisplayName = "当日比較値 (Open,Close,High,Low,Volume等) "];
```

```
Var:
    double RefValOfHizuke( 0.0 ),
    double Sagaku( 0.0 ),
    double Kairi( 0.0 );

    if Date[0] = Hizuke then RefValOfHizuke = RefVal[0];
    if LastBarOnChart then begin
        Sagaku = ComVal[0] - RefValOfHizuke;
        Kairi = 0.0;
        if RefValOfHizuke <> 0.0 then Kairi = Sagaku / RefValOfHizuke ;
        Plot1( Hizuke, "基準日" );
        Plot2( RefValOfHizuke, "基準値" );
        Plot3( ComVal[0], "比較値" );
        Plot4( Sagaku, "差額" );
        Plot5( Kairi, "変動率%" );
    end;
```

```
//上下色分け
if Plot4 > 0 then begin SetPlotColor( 4, UpColorDefault );
end else if Plot4 < 0 then SetPlotColor( 4, DownColorDefault );
if Plot5 > 0 then begin SetPlotColor( 5, UpColorDefault );
end else if Plot5 < 0 then SetPlotColor( 5, DownColorDefault );
```

- ※ 入力項目は基準日でEL形式YYYY MM DDになっています、最初のYは2000年以降なら1、1999年以前なら0です
- ※ 過去に遡る日にちが多い時は、設定の「累積計算のためのデータをロード」をチェックしてロードする追加バーが必要です
- ※ 遡る日にちに応じて入力してください、足りないとは基準値と変動率がゼロになります



外部データCSV（読込）

フォーラムを見て参考になったので外部データ（csv）からデータを取り込むインジケータを作成しました

殆どココアさんのコピペですがw 色々汎用性があると思います 参照トレステフォーラム 99, 117

まずこもりばんさんの所からcsvファイルを有難くDLして必要なデータに加工します <https://hesonogoma.com/stocks/japan-all-stock-prices.html>

トレステ側とcsvファイル側ではエンコードが違うので文字化けしてしまいます

タブ区切りやスペース区切りでの読込みも分からなかったので今回はカンマ区切りでエンコードします

プログラムの方でエンコード出来るかも知れませんがこれもやり方が分かりませんでした！(´Д`*)

・エンコード手順

エクセルで開き、ファイル-名前をつけて保存-csv（カンマ区切り）-ツール-Webオプション-エンコード-Unicode（UTF-8）で保存

又はメモ帳で開き、ファイル-名前をつけて保存-文字コードをUnicodeかUTF-8で保存（どちらでも可）

The screenshot shows Microsoft Excel with a CSV file open. The spreadsheet contains financial data for 'SC' (有利子負債(百万円)). A '名前を付けて保存' (Save As) dialog box is open, showing the file name 'japan-all-stock-financial-results.csv' and the file type 'CSV (カンマ区切り) (*.csv)'. The 'Web オプション' (Web Options) dialog box is also open, showing the 'エンコード' (Encoding) tab. The current encoding is 'Unicode (UTF-8)', and the option '常に Web ページを既定のエンコードで保存する(A)' (Always save Web pages in the default encoding) is checked.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	SC	有利子負債(百万円)											
2		1301	50919										
3		1332	207749										
4		1333	272208										
5		1352	19140										
6		1376	-										
7		1377	4192										
8		1379	29978										
9		1380	1852										
10		1381	35										
11		1382	-										
12		1383	1394										
13		1384	2708										
14		1400	-										
15		1401	316										
16		1407	31729										
17		1408	1922										
18		1413	9112										
19		1414	-										
20		1417	17322										
21		1418	4384										
22		1419	31317										
23		1420	23590										
24		1429	834										
25		1430	3054										
26		1431	-										
27		1433	650										
28		1434	3326										
29		1435	-										
30		1436	831										
31		1438	374										
32		1439	337										
33		1491	588										
34		1514	2690										
35		1515	23813										
36		1518	14817										
37		1605	687684										
38		1606	36010										
39		1662	141903										
40		1663	1330										
41		1711	611										
42		1712	2524										
43		1716	118										
44		1717	-										

※今回分からなかった事が分かったら後で修正します

外部データCSV（読込）（2）

コードと必要な項目で全銘柄のcsvファイルを2つ分けて作り、有利子負債と優待内容のインジケータを作ってみました

銘柄コード	銘柄名	有利子負債 (百万円)	優待内容
1 2915	ケンコーマヨネーズ	2409	100株以上 1000円相当自社製品
2 6238	フュー	-	100株以上 2000円相当のオリジナル・クオカード
3 7164	全国保証	-	100株以上 3000円相当のクオカード
4 7942	JSP	14606	100株以上 3000円相当の社会貢献寄付金付オリジナル・クオカード
5 8591	オリックス	4138451	100株以上 自社取引先取扱商品等カタログギフト3月のみ
6 8697	日本取引所グループ	52433	100株以上 1000円相当のクオカード
7 8771	イー・ギャランティ	5	100株以上 1500円相当のクオカード

TradeStationで作成。©TradeStation Technologies, Inc. All rights reserved./無断複写・転載を禁じます。

カウントで確認したら複数の項目でもいけそうでした、少し試した程度ですが全銘柄に全項目埋まっていないとダメなのかなという感じてしたCSVで読み込んだ値は文字型になるので、数値型に変換する場合はParseを使用して変換する

エクセル側でVLOOKUPしたり、コード順にトレスト側とエクセル側を合わせるといった作業もいらないのでとても楽です
頻繁に内容が変わらない項目や銘柄増減以外なら更新頻度それほど必要ないのでメンテも楽です

● プログラム

```
using elsystem;  
using elsystem.io;  
using tsdata.common;
```

```
inputs:                                     //任意の場所、ファイル名でOK  
    FileName("C:\Program Files (x86)\TradeStation 9.5\MyWork\japan-all-stock-financial-results.csv");
```

```
variables:  
    string MyCode( "" ),  
    double MyValue( 0 ),  
    string Code4( "" ),  
    StreamReader SR( NULL ),                //ファイルから文字を読取るクラスを実装、書き込み例については「StreamWriter」を参照  
    TokenList TL( NULL );                  //フィールド名のリスト(トークンリスト)に基づいてコレクションを作成する基本クラス
```

```
once begin  
    try SR = StreamReader.create(FileName); //例外発生テストに使用（読取ファイルの完全パスを指定新規インスタンスを初期化）  
    catch (FileNotFoundException ex)      //指定された例外に対応して、追加コードを実行するために使用  
    throw ex.create( "ファイルが見つかりません"); //イベントログに表示される例外のインスタンスを作成するために使用  
    end;{ try }  
    Code4 = symbol.Substring(0,4);         //エクセル側が4桁コードなのでシンボルコードから4桁だけ抜き出す  
end;{ once }
```

```
while Code4 <> MyCode and SR.EndOfStream = false begin //現在のファイル位置がファイルの末尾かどうかを示す値を取得  
    TL = TokenList.Create( SR.ReadLine()); //クラスの新規インスタンスを指定名に対して初期化（ファイルから文字の行を読取り）  
    if TL[0] = Code4 then begin  
        MyCode = TL[0];  
        if TL[1] <> "-" then MyValue = Double.Parse( TL[1]); //ハイフンを数値に変えるとエラー表示になるので除外  
        Plot1( MyValue, "有利子負債"); //CSVで読み込んだ値はstring型になるので、数値のDouble型に変換  
    end;  
end;{ while }
```

```
SR.Close(); //StreamReaderオブジェクトを閉じ、リーダーに関連付けられたシステムリソースを解放
```

決算発表日

決算発表のあった企業をスキャナーを使って日付でピックアップするために作成しました、中身的には営業進捗の直近決算日と同じです
営業進捗のインジケータでスキャンすると頻繁にトレストゲがクラッシュするので日付のみのインジケータで試作
クラッシュしなかったら営業利益をスキャンしてる時にクラッシュすることも分かるので！クラッシュする原因はゆるく追及していきます
なぜかスキャナーで日付を20170804等限定すると「選択された基準を満たす銘柄コードはありませんでした」になるので（EL形式でも同じ）
全銘柄表示させて日付で昇順/降順で並べ替えて必要分ピックアップの方がスキャンも速いと思います
今のところ実績データが作成されるのは発表日の夜中から翌朝です、朝起きた頃には作成されてるでしょう

● プログラム

```
using tsdata.marketdata;
using elsystem;

vars:
    FundamentalQuotesProvider FQP( null ),
    DateTime Kessan( null );
constants:
    string KeyActualFieldName( "CR_Op" );
method bool GetQuoteVal
( FundamentalQuote fq, DateTime tempdt, out DateTime Kessan )
variables:
    int fqCount,
    bool QuoteFound,
    DateTime fqDateTime;
begin
    if fq = null or tempdt = null then return false;
    QuoteFound = false;
    Kessan = DateTime.Create( 1900, 1, 1 );
    for fqCount = fq.Count - 1 downto 0 begin
        fqDateTime = fq.PostDate[fqCount];
        if tempdt >= fqDateTime and fqDateTime >= Kessan then begin
            Kessan = fq.PostDate[fqCount];
            QuoteFound = true;
        end;
    end;
    return QuoteFound;
end;

once begin
    FQP = new FundamentalQuotesProvider();
    FQP.Symbol = Symbol;
    FQP.Fields += KeyActualFieldName;
    FQP.LoadProvider();
end;

if GetQuoteVal( FQP[KeyActualFieldName], BarDateTime, Kessan ) then
    if Kessan <> 0 then Plot1( Kessan.ToString(), !( "直近決算日" ) );
```

IPO初値

普段IPO銘柄あまり触る事はないけど、大物IPOメルカリの相場きっかけで今年のIPOはどんな感じが、掘り出し物はないかと気になったので、また明石の入道さんのワークスペースのDLがまだ出来なかったので自作しちゃいましたw
1つでも良かったけど初値と公募に分けましたが特に意味はありません

● プログラム

```
Value1 = BarNumber;
if Volume[BarNumber] = 0 then Value1 = Value1 - 1; //ゴミデータ対応

Value2 = ( DateToJulian( CurrentDate ) - DateToJulian( Date[Value1] ) );{経過日計算 (土日祝日含む) }

if Value2 = 0 then begin Value3 = DailyOpen; //上場初日初値 ※ 上場初日は日足だとデータ不足エラーで表示されないで分足に設定する
end else if Value2 = 1 then begin Value3 = OpenD(1); //二日目の日足が確定しないと[BarNumber]の1が確定しない感じなので昨日の始値で対応
end else if Value2 >= 2 then begin Value3 = Open[Value1]; //二日目以降の対応
end;

Plot1( Value3, "初 値" );
if Plot1 <> 0 then begin
  if Close - Plot1 > 0 then Plot2( Close - Plot1, "現在差", UpColorDefault );
  if Close - Plot1 < 0 then Plot2( Close - Plot1, "現在差", DownColorDefault );
  if Close / Plot1 > 1 then Plot3( ( Close / Plot1 - 1), "騰落率", UpColorDefault );
  if Close / Plot1 < 1 then Plot3( ( Close / Plot1 - 1), "騰落率", DownColorDefault );
end;
Plot4( Date[Value1], "初値日" );
Plot5( Year( CurrentDate ) - Year( Date[Value1] ), "経過年" );
Plot6( ( Value2 ), "経過日" );{土日祝日含む} //Plot6( Value1, "経過日" ); {営業日ベース}
```

- ※ 初値も公募も累積計算のため追加データをロードをチェックし遡って表示させたい日数以上の数字を設定して下さい
目安は1か月20日として1年分なら240以上
- ※ 初値日、上場日、ロックアップ解除日は日付、騰落率はパーセントにプロパティのスタイルで設定して下さい
チャートに挿入する時はスケージングのスケール位置は元データに軸を合わせるに設定して下さい
開発の検証時に設定すればインジ挿入時毎回設定しなくて済みます
- ※ 上場日前に公募価格で出来高ゼロの[BarNumber]も付与されてるデータが数件あったので出来高ゼロは除いています
- ※ 上場初日は日足だとデータ不足エラーで表示されないで、その銘柄のみ当日だけ分足、秒足、ティックに設定で表示されます
分足も2本目が確定したら表示されるようになるので、5分足の場合9時に寄ったとしても10分過ぎないと表示されません
すぐに表示させたい場合は秒足かティックがお勧めです
- ※ 二日目の日足が確定しないと[BarNumber]の1が確定しない感じなので、二日目は昨日の始値で表示しています
- ※ 営業日ベースの経過日数の方が良い方はコメントの部分と入れ替えで表示できます //Plot6(Value1, "経過日"); {営業日ベース}
- ※ 経過年をキーにスキャナーで簡単にIPO銘柄をスキャンできます、2年、3年前と遡る時は累積計算のため追加データを忘れずに！



	A1	コード						
	A	B	C	D	E	F	G	H
1	コード	公募	分割	上場日				
3381	4379	0	1	1180101				
3382	4380	1240	1	1180223				
3383	4381	2200	1	1180404				
3384	4382	4500	1	1180420				
3385	4383	0	1	1180101				
3386	4384	1500	1	1180531				
3387	4385	3000	1	1180619				
3388	4386	2000	1	1180621				

※ 画像はIPO公募用のcsvファイル

IPO公募

● プログラム

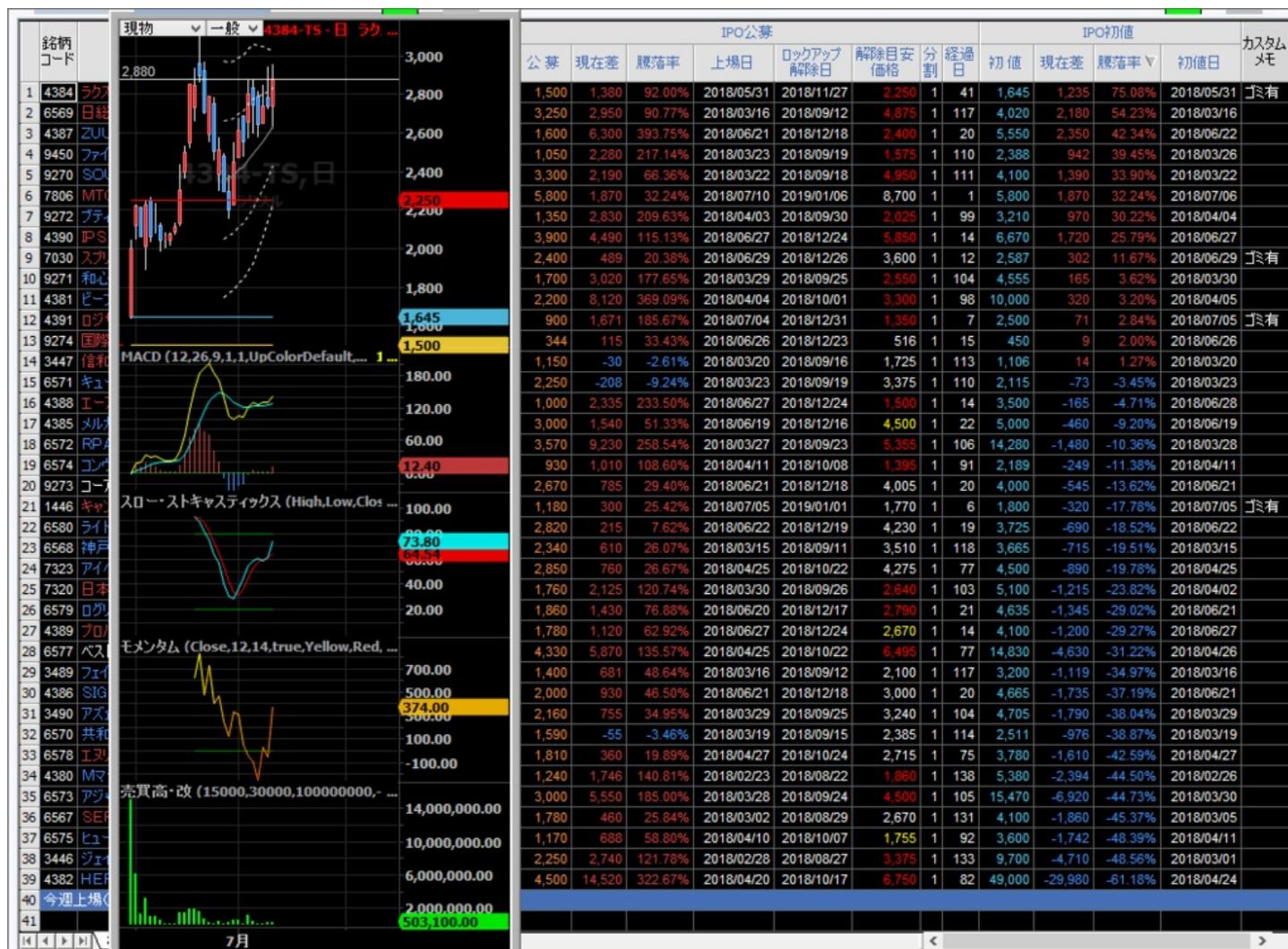
```
using elsystem;
using elsystem.io;
using tsdata.common;
inputs:
    FileName("C:\Program Files (x86)\TradeStation 9.5\japan-all-stock-financial-ipo.csv"),
    int RockUp_Date( 180 ) [DisplayName = "ロックアップ期間"],           {銘柄毎に変更ある場合は設定}
    double RockUp_Rate( 1.5 ) [DisplayName = "ロックアップ解除価格"];    {銘柄毎に変更ある場合は設定}
variables:
    string MyCode(""),
    double MyVal1( 0 ),           //公募価格
    double MyVal2( 0 ),           //分割
    double MyVal3( 0 ),           //上場日
    string Code4(""),
    StreamReader SR( null ),       //ファイルから文字を読み取るクラスを実装します、書き込み例については「StreamWriter」を参照
    TokenList TL( null );         //フィールド名のリスト (トークンリスト) に基づいてコレクションを作成する基本クラスです

once begin
    try SR = StreamReader.create( FileName );           //プログラム実行中の異常な状態、例外の発生をテストするために使用されます (読み取るファイルの完全パスを指定して新規インスタンスを初期化します)
    catch ( FileNotFoundException ex )               //指定された例外に対応して、追加コードを実行するために使用されます
    throw ex.create( "ファイルが見つかりません" );    //イベントログに表示される例外のインスタンスを作成するために使用します
    end;{ try }
    Code4 = symbol.Substring( 0,4 );                   //エクセル側が4桁のコードなのでシンボルコードから4桁だけ抜き出します
end;{ once }

while Code4 <> MyCode and SR.EndOfStream = false begin //現在のファイル位置がファイルの末尾かどうかを示す値を取得します
    TL = TokenList.Create( SR.ReadLine() );           //クラスの新しいインスタンスを指定名に対して初期化します (ファイルから文字の行を読み取り、データを文字列として返します)
    if TL[0] = Code4 then begin
        MyCode = TL[0];
        MyVal1 = Double.Parse( TL[1] );               //公募価格
        MyVal2 = Double.Parse( TL[2] );               //分割
        MyVal3 = Double.Parse( TL[3] );               //上場日
    end;
end;{ while }

Value1 = 1 / MyVal2;                                 //分割考慮計算
Plot1( MyVal1 * Value1, "公募");
if Close - Plot1 > 0 then Plot2( Close - Plot1, "現在差", UpColorDefault );
if Close - Plot1 < 0 then Plot2( Close - Plot1, "現在差", DownColorDefault );
if Close / Plot1 > 1 then Plot3( ( Close / Plot1 - 1), "騰落率", UpColorDefault );
if Close / Plot1 < 1 then Plot3( ( Close / Plot1 - 1), "騰落率", DownColorDefault );
Plot4( MyVal3, "上場日");
Value2 = ( DateToJulian( MyVal3 ) + RockUp_Date );    //ロックアップ期間計算
Plot5( DateToString( Value2 ), "ロックアップ解除日");
Plot6( MyVal1 * Value1 * RockUp_Rate, "解除目安価格");
if last > ( Plot6 * 1.1 ) then begin SetPlotColor( 6, Red ); //ロックアップ価格以上黄色、10%以上赤色
end else if last - Plot6 > 0 then SetPlotColor( 6, Yellow );
Plot7( MyVal2, "分割");
Value3 = ( DateToJulian( Currentdate ) - DateToJulian( MyVal3 ) ); {経過日計算 (土日祝日含む)}
Plot8( Value3, "経過日" );                            {土日祝日含む}
if Value3 <= RockUp_Date then begin
    if RockUp_Date * 0.95 <= Value3 then begin SetPlotColor( 5, Red ); //期間中ロックアップ期間90%以上黄色、95%以上赤色
    end else if RockUp_Date * 0.9 <= Value3 then SetPlotColor( 5, Yellow );
    if RockUp_Date * 0.95 <= Value3 then begin SetPlotColor( 8, Red ); //期間中ロックアップ期間90%以上黄色、95%以上赤色
    end else if RockUp_Date * 0.9 <= Value3 then SetPlotColor( 8, Yellow );
end;
SR.Close(); //StreamReader オブジェクトおよび基礎ファイルを閉じ、リーダーに関連付けられたシステムリソースを解放します
```

IPO公募 (2)



※ 任意の場所、ファイル名でcsvファイルを用意します、その内容を読み取りレーダースクリーンやチャートに表示します

内容は証券コード、IPO銘柄の公募価格、分割数、上場日 (EL日付のYYYYMMDD形式) です
 必要な証券コードのみ用意でも可ですが、レーダースクリーン等で、ないコードを入力した場合エラーになる可能性もあるので一応1000-9999まで用意しておいた方が無難だと思います

1行目は証券コード、公募価格、分割、上場日などの見出しが入っていても大丈夫です

※ 用意した銘柄コードの全項目が埋まるように空白が無いようにcsvファイルを作成して下さい
 上場日も用意した証券コード全てに1/1などで適当に埋めきってから必要なコードの部分の正しい上場日に変更して下さい (なるべくStreamReaderで読み取りエラーが出ないようにするため)

※ 分割数は全銘柄デフォルトは1で設定して下さい1が分割がない状態です、1 : nで分割されたらnの数字に変更して下さい。2分割の場合は1から2へ変更、以降は乗算していきます。例えば2分割後の3分割は6へ変更 (1/nを公募価格に乗算しているため)

※ StreamReaderで読むとcsvファイルが読み取り専用になってしまい次回編集する時めんどくさくなるので、エクセルで編集後、csvファイルを作成、任意の場所に上書きコピーして下さい

※ csvファイルのエンコードとかその他よく分からない場合は、c s v読み込みのところを見て下さい

※ ロックアップ期間と解除価格の倍率は入力項目です、デフォルト180日と1.5倍にしてあるので個別に異なる場合は銘柄毎変更して下さい

※ 価格や期間の色の変更はプログラムで固定でやっているので好みで変更して下さい

※ 初値、公募等レーダースクリーンとチャートで色を合わせると分かりやすくなります

ファンダメンタルデータ（検証用）

リーダースクリーン用です

既存のカスタムファンダのインジケータでも一応使えますが1つずつ確認するのは面倒なので前に作ったものを改良しました

ファンダメンタルのデータはヘルプの日本株と書かれているものを見て下さい、書いてないものは米国版で現在の日本のトレストでは使用できません

入力項目は銘柄コード、項目、遡りでデフォルトは、0000-T S、CR-OP（実績営業利益）、0、です

画面には指定した項目、遡りの値を表示（銘柄コードは指定しなくても表示します）

銘柄コードを入力したものは印刷ログに全データを表示します

既存のインジケータの多くは入力項目の、月数、連結単独、会計基準等でデータを選択して表示していますが、隠された項目も表示できるようになってますのでデータベースの概要を掴みやすいと思います

まだ隠されてる項目もあるかもしれませんが今現在見つけたものだけ表示しています

例は7806MTGです

営業利益の実績は2件、最新のデータは5289で7/10に作成されていますが既存のインジのデータは5787になっています

これは最新のデータは月数が6で半期の実績の為というのがわかります

隠された項目でプログラム内で選択されているため実績のインジには1つ前の月数12の5787が表示されています

	銘柄コード	銘柄名	営業利益		ファンダメンタルデータ(検証用)						
			実績	予想	Count	Value	Date	Month	Acct	Consol	TermType
1	7806-TS	MTG	5,787	7,547	2	5289.0	2018/07/10	6	JSTD	Consolidated	MidTerm
2	3447-TS	信和	2,306	2,370	3	2306.0	2018/05/15	12	IFRS	Consolidated	FullTerm
3	7030-TS	スプリックス	1,164	2,351	2	1504.0	2018/06/29	6	JSTD	NonConsolidated	MidTerm
4	6569-TS	日総工産	1,800	2,099	4	1796.0	2018/05/10	12	JSTD	NonConsolidated	FullTerm
5	6571-TS	キュービーネットホー	1,502	1,714	3	183.0	2018/05/15	3	IFRS	Consolidated	QuarterTerm
6	3489-TS	フェイスネットワーク	1,237	1,700	3	1237.0	2018/05/14	12	JSTD	NonConsolidated	FullTerm
7	9270-TS	SOU	1,030	1,693	4	267.0	2018/07/13	3	JSTD	Consolidated	QuarterTerm
8	9273-TS	コア商事ホールデ	1,614	1,322	2	884.0	2018/06/21	9	JSTD	Consolidated	QuarterTerm
9	6568-TS	神戸天然物化学	1,222	1,300	3	1222.0	2018/05/14	12	JSTD	NonConsolidated	FullTerm
10	4390-TS	IPS	902	921	1	902.0	2018/06/27	12	JSTD	Consolidated	FullTerm

検索

開発 / 先物II / PF

7806-TS
0 CR_OP : 5289.0 2018/07/10 6 JSTD Consolidated MidTerm
1 CR_OP : 5787.0 2018/03/01 12 JSTD Consolidated FullTerm
CR_OP Operating Income doubleval

印刷ログ

- プログラム

```
using elsystem;
```

```
using tsdata.marketdata;
```

```
inputs: { 入力変数宣言 }
```

```
string FundSymbol( "0000-TS" ) [DisplayName = "Symbol"], //任意の銘柄コードを指定  
string FundField( "CR_OP" ) [DisplayName = "Field"], //任意の項目を指定：実績営業利益  
int PeriodsAgo( 0 ); //任意の遡りを指定：0が最新
```

```
vars: { 変数宣言 }
```

```
FundamentalQuotesProvider FQP( null ), //FundamentalQuotesProvider クラス  
bool logPrint( true ); //印刷ログ出力
```

ファンダメンタルデータ（検証用）（2）

```
constants: { 定数宣言 }
    string KeyMonthsReported( "MonthsReported" ),           //月数
    string KeyAccounting( "AccountingStandard" ),           //会計基準
    string KeyConsolidated( "ConsolidationLevel" ),         //連単
    string KeyTermType( "TermType" );                       //期間タイプ

method void Fundamental_Updated( elsystem.Object sender, tsdata.marketdata.FundamentalQuoteUpdatedEventArgs args )
vars: int fieldNo, int counter, string output;
begin
//入力した銘柄コード、項目、遡り、データを画面に表示
    if FQP.HasQuoteData( 0 ) then begin
        Plot1( FQP.Quote[0].Count, "Count" );
        Plot2( FQP.Quote[0].Value[PeriodsAgo].ToString(), "Value" );
        Plot3( FQP.Quote[0].PostDate[PeriodsAgo].ToString(), "Date" );
        Plot4( FQP.Quote[0].ExtendedProperties[PeriodsAgo][KeyMonthsReported].ToString(), "Month" );
        Plot5( FQP.Quote[0].ExtendedProperties[PeriodsAgo][KeyAccounting].ToString(), "Acct" );
        Plot6( FQP.Quote[0].ExtendedProperties[PeriodsAgo][KeyConsolidated].ToString(), "Consol" );
        Plot7( FQP.Quote[0].ExtendedProperties[PeriodsAgo][KeyTermType].ToString(), "TermType" );
    end;
//全データを印刷ログに表示
    if FundSymbol.Contains( symbol ) then begin                //指定名を持つクオートがプロバイダーに存在する場合は真
        if logPrint then print( Symbol );
        if FQP.HasQuoteData( 0 ) then begin                    //指定インデックスを持つクオートに値が含まれる場合は真
            for counter = 0 to FQP.Quote[0].Count -1 begin
                output = NumToStr( counter, 0 );
                output += " ";{ Tab }
                output += " " { Space } + FQP.Fields[0];
                output += " ";{ Tab }
                output += " " + FQP.Quote[0].Value[counter].ToString();
                output += " ";{ Tab }
                output += " " + FQP.Quote[0].PostDate[counter].ToString();
                output += " ";{ Tab }
                output += " " + FQP.Quote[0].ExtendedProperties[counter][KeyMonthsReported].ToString();
                output += " ";{ Tab }
                output += " " + FQP.Quote[0].ExtendedProperties[counter][KeyAccounting].ToString();
                output += " ";{ Tab }
                output += " "{ Space } + FQP.Quote[0].ExtendedProperties[counter][KeyConsolidated].ToString();
                if FQP.Quote[0].ExtendedProperties[counter][KeyConsolidated].ToString() = "NonConsolidated" then
                    output += " "{ Tab } else output += " ";{ Tab*2 }
                output += " " + FQP.Quote[0].ExtendedProperties[counter][KeyTermType].ToString();
                if logPrint then print( output );
            end;{ For }
            if logPrint then print( FQP.Fields[fieldNo], " ", FQP.Quote[fieldNo].Description, " ", FQP.Quote[fieldNo].Type.ToString() + NewLine );
        end else print( FQP.Fields[fieldNo], " is not available" + NewLine );
    end;{ FundSymbol.Contains }
end;{ Fundamental_Updated }

once begin
    FQP = FundamentalQuotesProvider.Create();
    FQP.Symbol = symbol;
    FQP.Fields += FundField;
    FQP.Updated += Fundamental_Updated;
    FQP.Load = true;
end;{ once }
```

※ " ";{ Tab }のWクォーテーションの中はタブで*2はタブ2回です、" ";{ Space }のWクォーテーションの中は半角スペースです
多分コピーでT a bとスペースおかしくなると思うので入力し直してください

※ この表示の印刷ログのフォントはMSゴシックで揃えていますので自身のフォントで使用する場合はスペースやタブで見やすく調整して下さい

データの共有

いよいよアプリ開発

EL勉強し始めて6ヶ月、主にレーダースクリーンのインジケータ作成をしてきたがいよいよアプリ開発へ

取り敢えずヘルプを見ながらソースの解析から始めようとしたが、参考になるソースが...

ファイル - 開く (アプリ) で分かる通り、マネックスアルゴ、マルチチャート、マルチクオートの3つしかないw

中身はELで書かれているので何とか理解できると思うが、ステップ数が長くてだるい(泣;)あまりに長くそっと閉じた...

1度挫折を経て、nariさんのブログを見直しなんとか頑張れそうと立ち直る(泣) <http://nari-trade.blog.so-net.ne.jp/>

解説付きで何本かアプリ作成されてたのでとても参考になりました！nariさんありがとうございます！

あとヘルプにあったサンプルも参考になりました！参考元 `!ex_DataGridViewSample` リンク

http://help.tradestation.com/09_05/Monex/jpn/tsdevhelp/Subsystems/elobject/example/!ex_datagridviewsample.eld

アプリを作るに当たってデータの共有について少し調べてみました

データ共有 (どの共有も一時メモリのな感じでトレストを起動中のみ可能、トレストを終了させるとクリアされてしまう)

- Dictionary クラス (コレクション) 名前空間: `elsystem.collections`

キー/値のペアのオブジェクトを定義し、使用する値を保存、共有を可能にする

異なる銘柄で値が共有できる (1つの分析テクニック、ストラテジー、アプリ内に限る)

初期化: `Create`、追加: `Add(key,value)`、削除: `Remove(key)`、確認: `Contains(key)`、リセット: `Clear`

- Global Dictionary クラス (コレクション) 名前空間: `elsystem.collections`

Dictionaryに更に加えて異なる分析テクニック、ストラテジー、アプリ間で共有可能、異なるウインドウでも可能

但し共有の仕方で初期化の設定方法が異なる

同一ウインドウの場合、`myGlobal = GlobalDictionary.create();`

異なるウインドウの場合、`myGlobal = GlobalDictionary.create(true, "share_name");`

`Create()`、`Create(share,name)` 設定に応じて初期化も2種類、あと追加・変更・削除・消去に応じてイベントハンドラーが使用可能

- GlobalValue クラス (コンポーネント) 名前空間: `elsystem`

変数名/値で定義、異なる分析テクニック、ストラテジー、アプリ間で共有可能、異なるウインドウでも可能

作成はツールボックスのコンポーネントから

`Create`: 初期化、`Load`: オブジェクトをロード、`Type`: オブジェクトのタイプを取得

読み書きには、値のタイプによってプロパティが必要 (`DateValue`、`DoubleValue`、`IntegerValue`、`StringValue`)

1つの名前につき1つの値、名前をキーに値をタイプ指定で保存共有な感じ、書込みは `名前.値のプロパティ`、読み込みは `名前.Load`

変更(`Updated`)に応じてイベントハンドラー使用可能

オブジェクトをロードすれば普通の変数のように扱える

例: `GlobalValue1.DoubleValue = 123.45;` 例: `MyPrice = GlobalValue1.DoubleValue;`

- 配列 `Array` (予約語) `Arrays` と同じ

使い次第で異なる銘柄で値が共有するように使用する事もできる (1つの分析テクニック、ストラテジー、アプリ内に限る)

10次元までの多次元配列を定義可能、配列名は最大20文字まで!

各インデックス番号にスペースを割り当てるため、必要以上に大きな配列の宣言は避ける

配列に関する予約語や関数は沢山あるので、ヘルプで「配列 and Reserved Word」や「配列 and Function」等で検索してみてください

例: 1次元配列 配列名[要素数] (初期値);

`Array: MyYuutaiArray[9999] ("");` 銘柄コードに対応した優待内容を格納

例: 2次元配列 配列名[1次元要素数,2次元要素数] (初期値);

`Array: MyNumericArray[200, 5] (0);` 200日分の日付、始値、高値、安値、終値を格納して、移動平均や最高値、最安値等を計算可能

サンプルプログラム

ヘルプにあったプログラムを元にサンプルプログラムを作成してみました、オブジェクト指向で！
今回は只の雛型プログラムです、これをテンプレに必要な項目を選び肉付けすれば簡単なアプリが作成できると思います
アプリのパーツや型の感じがつかめると思うので色々試してみてください（使用の際には表示-EL印刷ログを表示して下さい）

あとは大きさ、数、色、レイアウトや細かい設定等、クリックした項目に応じて必要な処理等、エラー対応等、ELで作成すればOKです
アプリとインジケータやショーミー等の一番の大きな違いは自由な枠組みを作れるかどうか！そこだけだと思いました（今のところw）
自由なだけにレイアウトとかセンスが問われますが、でも基本殆どがエクセル型ですw
色んな情報を表示出来るのがレーダースクリーン、売買実績を表示でタイム&セールス、板と出来高を表示でマトリクスって感じですが
気付くと型的には殆どエクセル型と同じですが、表示するデータによって随分違うアプリに感じていた気がします(■°o°■)

ヘルプのアプリは、イニシャライズイベントにフォームやグリッドの設定を細かく出来ていました
フォームはEL開発環境で右クリック-フォーム追加でイニシャライズイベントの中に入れる事が出来ましたが、細かい変更や設定が分かりませんでした
コンポーネントもチェックが入ってるにも関わらずWinformの所には出てこないし、良くわかりません(´д`；)

興味のある人は、ヘルプアプリのジェネレーションコードを表示してみるとフォームやグリッドが細かく設定されてるのが分かります
今回は殆ど設定していませんが、細かく設定しようとするればフォームとグリッドの設定だけで更に100ステップ弱必要という事になります
参考元 lex_DataGridViewSampleリンク

http://help.tradestation.com/09_05/Monex/jpn/tsdevhelp/Subsystems/elobject/example/lex_datagridviewsample.eld

オブジェクト指向プログラミングは開発手法の1つ！
人それぞれの理解度と解釈で、色々な比喻や例えで余計分かりにくくしてる感(°_°)

オブジェクト指向 (object oriented)
オブジェクト指向プログラミングは、「オブジェクト（もの）中心に考えるソフトウェア開発手法」
ソフトウェア全体として機能を実現するだけでなく、保守性や再利用性に配慮して、個々の部品の独立性も重視する

従来の開発手法は、「機能中心」最初に全体として実現する機能を定義し、徐々に細分化していく
「構造化分析/設計開発手法」として体系化されて長い間主流として使われてきた

● プログラム

```
using elsystem; //クラス参照宣言 //tsdataクラスによって使用される基本クラスと、その他一般的なelsystemクラス
using elsystem.windows.forms; //フォームのコントロールおよびコンテナの作成に使用されるクラス
using elsystem.drawing; //フォームコントロールやドローイングオブジェクトの色およびフォント特性を記述するために使用されるクラス

variables: //変数宣言 フォームや表、タブ、ボタン、ボックス、辞書等の宣言
    Form Form1( null ), //フォーム Formクラス
    TabControl Tab( null ), tabPage TPage1( null ), tabPage TPage2( null ), //タブ TabControlクラス
    DataGridView DGV1( null ), //表 DataGridViewクラス
    DataGridViewButtonColumn BtnCol( null ), //ボタン DataGridViewButtonColumnクラス
    DataGridViewCheckBoxColumn ChkBoxcol( null ), //チェックボックス DataGridViewCheckBoxColumnクラス
    DataGridViewComboBoxColumn ComBoxcol( null ), //コンボボックス DataGridViewComboBoxColumnクラス
    DataGridViewDateTimePickerColumn DTPcol( null ), //日時ピッカー DataGridViewDateTimePickerColumnクラス
    DataGridViewLinkColumn Linkcol( null ), //リンク DataGridViewLinkColumnクラス
    DataGridViewNumericUpDownColumn NUDcol( null ), //数値アップダウン DataGridViewNumericUpDownColumnクラス
    DataGridViewTextBoxColumn Tboxcol( null ), //テキストボックス DataGridViewTextBoxColumnクラス
    DataGridViewColumn Textcol( null ), //テキスト DataGridViewColumnクラス
    DataGridViewRow Row1( null ), //行 DataGridViewRowクラス
    DataGridViewColumn Col1( null ); //列 DataGridViewColumnクラス
{ ----- } //※ 検証する時に、プロパティでイニシャライズイベントに設定
method void AnalysisTechnique_Initialized( elsystem.Object sender, elsystem.InitializedEventArgs args )
begin
    Clearprintlog(); //印刷ログを削除
    SelectGrid(); //パーツの設定・選択
    Form1.Show(); //フォームを表示
end;{ method AnalysisTechnique_Initialized }
```

サンプルプログラム (2)

```
{ ----- } //パーツの設定・選択
method void SelectGrid()
begin
//パーツの初期設定・表示
    Form_Column();           //フォーム
    Tab_Column();           //タブ
    DGV_Column();          //表
    Button_Column();       //ボタン
    Checkbox_Column();     //チェックボックス
    Combobox_Column();     //コンボボックス
    DateTimePicker_Column(); //日時ピッカー
    Linkcol_Column();      //リンク
    NumericUpDown_Column(); //数値アップダウン
    Textbox_Column();      //テキストボックス
    Text_Column();         //文字、数値
    Col1_Column();         //列
//列 見出し例
    DGV1.Columns.Add( "例 1 " );           //見出し例 1
    DGV1.Columns.Add( "列 1 " );          //列追加
    for Value1 = 0 to 9 begin             //9列処理
        if Value1 <= 1 then begin
            Row1_Column();                 //行 初期設定
            Row1.Cells[0].Value = "Button " + NumToStr(Value1, 0);
            Row1.Cells[1].Value = True;
            Row1.Cells[2].Value = "None";
            Row1.Cells[2].ToolTipText = "選択してちょ！";
            Row1.Cells[3].Value = DateTime.Create(2017, 8, Value1);
            Row1.Cells[4].Value = "Link " + NumToStr(Value1, 0);
            Row1.Cells[5].Value = Value1;
            Row1.Cells[6].Value = "サンプルだよ";
            Row1.Cells[7].Value = Value1;
            Row1.Cells[8].Value = Value1;
            Row1.Cells[9].Value = "プリントログ削除";
        end;
    end;
end; { SelectGrid }
{ ----- } //フォーム初期設定
method void Form_Column()
begin
    Form1 = Form.Create();                 //フォームの新規インスタンスを初期化
    Form1.Dock = DockStyle.Fill;          //フォームのコントロールがドッキングされる位置および方法を指定
end; { Form_Column }
{ ----- } //タブ初期設定
method void Tab_Column()
begin
    Tab = TabControl.Create();             //TabControlクラス、TabControlの新規インスタンスを初期化
    Form1.AddControl( Tab );               //フォームにタブのコントロールを追加
    Tab1_Column();                          //タブ 1 処理
    Tab2_Column();                          //タブ 2 処理
    Tab.Dock = DockStyle.Fill;              //コントロールがドッキングされる位置および方法を指定 ヘルプDockStyle列挙参照
end; { Tab_Column }
```

サンプルプログラム (3)

```
{ ----- } //タブ1 処理
method void Tab1_Column()
begin
    TPage1 = TabPage.Create( "タブ名 1",0,0);           //TabPageクラス、新規インスタンスを初期化 ※ タブの横、高さ機能していない
    Tab.AddControl( TPage1 );                         //TabControlクラス、TPage1のコントロールをタブに追加
end; { Tab1_Column }
{ ----- } //タブ2 処理
method void Tab2_Column()
begin
    TPage2 = TabPage.Create( "タブ名 2",0,0);           //TabPageクラス、新規インスタンスを初期化 ※ タブの横、高さ機能していない
    Tab.AddControl( TPage2 );                         //TabControlクラス、TPage2のコントロールをタブに追加
end; { Tab2_Column }
{ ----- } //表初期設定
method void DGV_Column()
begin
    DGV1 = DataGridView.Create();                     //DataGridViewクラス、DGV1の新規インスタンスを初期化
    TPage1.AddControl( DGV1 );                       //DGV1表をTPage1のコントロールに追加
    DGV1.Dock = DockStyle.Fill;                     //DGV1のコントロールがドッキングされる位置および方法を指定
    DGV1.AllowUserToAddRows = false;                //ユーザーがグリッド行を追加できる場合はtrue/false
    DGV1.ColumnHeadersDefaultCellStyle.Alignment   //ヘッダーのデフォルトのスタイルを取得または設定
    = DataGridViewContentAlignment.MiddleCenter;   //フォームコントロール内のテキストの整列
end; { DGV_Column }
{ ----- } //ボタン初期設定
method void Button_Column()
begin
    Btncol = DataGridViewButtonColumn.Create( "ボタン" ); //見出し
    Btncol.SortMode = DataGridViewColumnSortMode.Automatic; //ソート
    Btncol.AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;//列幅
    DGV1.Columns.Add( Btncol );
end; { Button_Column }
{ ----- } //チェックボックス初期設定
method void Checkbox_Column()
begin
    Chkboxcol = DataGridViewCheckBoxColumn.Create( "チェックボックス" );
    Chkboxcol.SortMode = DataGridViewColumnSortMode.Automatic;
    Chkboxcol.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
    DGV1.Columns.Add( Chkboxcol );
end; { Checkbox_Column }
{ ----- } //コンボボックス初期設定
method void Combobox_Column()
begin
    Comboxcol = DataGridViewComboBoxColumn.Create( "コンボボックス" );
    Comboxcol.SortMode = DataGridViewColumnSortMode.Automatic;
    Comboxcol.Items.AddRange( "None", "A", "B", "C", "D", "E" );
    Comboxcol.AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells;
    DGV1.Columns.Add( Comboxcol );
end; { Combobox_Column }
{ ----- } //日時ピッカー初期設定
method void DateTimePicker_Column()
begin
    DTPcol = DataGridViewDateTimePickerColumn.Create( "日時ピッカー" );
    DTPcol.SortMode = DataGridViewColumnSortMode.NotSortable; //※ ソート禁止！100%トレステクラッシュする
    DTPcol.AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;
    DGV1.Columns.Add( DTPcol );
end; { DateTimePicker_Column }
```

サンプルプログラム (4)

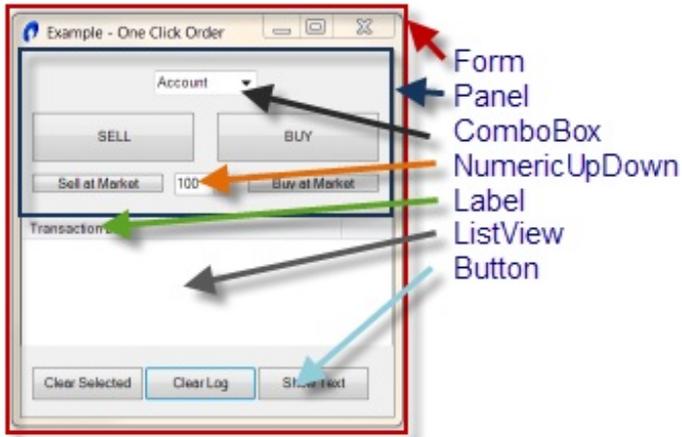
```
{ ----- } //リンク初期設定
method void Linkcol_Column()
begin
    Linkcol = DataGridViewLinkColumn.Create( "リンク" );
    Linkcol.SortMode = DataGridViewColumnSortMode.Automatic;
    Linkcol.AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells;
    DGV1.Columns.Add( Linkcol );
end; { Link_Column }
{ ----- } //数値アップダウン初期設定
method void NumericUpDown_Column()
begin
    NUDcol = DataGridViewNumericUpDownColumn.Create( "数値アップダウン" );
    NUDcol.SortMode = DataGridViewColumnSortMode.NotSortable; //※ ソート禁止！100%トレステクラッシュする
    NUDcol.AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells;
    DGV1.ColumnHeadersDefaultCellStyle.BackColor = elsystem.drawing.SystemColors.Control;
    DGV1.Columns.Add( NUDcol );
end; { NumericUpDown_Column }
{ ----- } //テキストボックス初期設定
method void TextBox_Column()
begin
    TBoxcol = DataGridViewTextBoxColumn.Create( "テキストボックス" );
    TBoxcol.SortMode = DataGridViewColumnSortMode.Automatic;
    TBoxcol.AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells;
    DGV1.Columns.Add( TBoxcol );
end; { TextBox_Column }
{ ----- } //テキスト初期設定
method void Text_Column()
begin
    Textcol = DataGridViewColumn.Create( "テキスト" );
    Textcol.SortMode = DataGridViewColumnSortMode.Automatic;
    Textcol.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
    Textcol.ReadOnly = True; //ユーザーが列のセルを編集できない場合はtrue/false
    DGV1.Columns.Add( Textcol );
end; { Text_Column }
{ ----- } //列初期設定
method void Col1_Column()
begin
    Col1 = DataGridViewColumn.Create( "" ); //DataGridViewColumnクラス、Colの新規インスタンスを初期化
    Col1.SortMode = DataGridViewColumnSortMode.Automatic; //列のソートモードを取得または設定
// DGV1.Columns.clear(); //列クリア
end; { Col1_Column }
{ ----- } //行初期設定
method void Row1_Column()
begin
    Row1 = DataGridViewRow.Create( "" ); //DataGridViewRowクラス、Rowの新規インスタンスを初期化
    DGV1.Rows.Add( Row1 ); //行追加
// DGV1.Rows.clear(); //行クリア
end; { Row1_Column }
```

サンプルプログラム (5)

```
{ ----- } //選択処理
//セルをクリックした時
method void DGV1_CellClick( elsystem.Object sender, elsystem.windows.forms.DataGridViewCellEventArgs args )
begin
    print("セルをクリック: 行=", args.RowIndex, " 列=", args.ColumnIndex);
    { ボタンを押した時 }
    if ( args.ColumnIndex = 0 and args.RowIndex >= 0 ) then begin
        print( DGV1.Rows[args.RowIndex].Cells[0].Text, " クリック", args.RowIndex, " 行の状態を印刷" );
        print( "チェックボックス選択 ", DGV1.Rows[args.RowIndex].Cells[1].Value.ToString() );
        print( "コンボボックス選択 ", DGV1.Rows[args.RowIndex].Cells[2].Text );
        print( "日時ピッカー選択 ", DGV1.Rows[args.RowIndex].Cells[3].Value.ToString() );
        print( "数値アップダウン選択 ", DGV1.Rows[args.RowIndex].Cells[5].Value.ToString() );
        print( "テキストボックス ", args.RowIndex, " ", DGV1.Rows[args.RowIndex].Cells[6].Value.ToString() );
        print( "テキスト ", DGV1.Rows[args.RowIndex].Cells[7].Value.ToString() );
        print( "例 1 ", DGV1.Rows[args.RowIndex].Cells[8].Value.ToString() );
        switch DGV1.Rows[args.RowIndex].Cells[2].Text begin
            case "None": print( "コンボNone処理 " ); break;
            case "A": print( "コンボ A 処理 " ); break;
            case "B": print( "コンボ B 処理 " ); break;
            case "C": print( "コンボ C 処理 " ); break;
            case "D": print( "コンボ D 処理 " ); break;
            case "E": print( "コンボ E 処理 " ); break;
        end;
    end;
    { 列9を押した時 }
    if ( args.ColumnIndex = 9 and args.RowIndex >= 0 ) then begin
        Clearprintlog(); //印刷ログを削除
    end;
end; { DGV1_CellClick }
//選択を変えた時
method void DGV1_SelectionChanged( elsystem.Object sender, elsystem.EventArgs args )
begin
    print( "選択変更: ", DGV1.SelectedCells.Count, " セル選択" );
end;
//値を変えた時
method void DGV1_CellValueChanged( elsystem.Object sender, elsystem.windows.forms.DataGridViewCellEventArgs args )
begin
    print( "値変更: 行=", args.RowIndex, " 列=", args.ColumnIndex );
end;
//列ヘッダーをクリックした時
method void DGV1_ColumnHeaderMouseClick( elsystem.Object sender, elsystem.windows.forms.DataGridViewCellMouseEventArgs args )
begin
    print( "列ヘッダーをクリック: 行=", args.RowIndex, " 列=", args.ColumnIndex);
end;
//行ヘッダーをクリックした時
method void DGV1_RowHeaderMouseClick( elsystem.Object sender, elsystem.windows.forms.DataGridViewCellMouseEventArgs args )
begin
    print( "行ヘッダーをクリック: 行=", args.RowIndex, " 列=", args.ColumnIndex);
end;
//ソートした時
method void DGV1_SortCompare( elsystem.Object sender, elsystem.windows.forms.DataGridViewSortCompareEventArgs args )
begin
    print( "ソート: 行=", args.RowIndex1, " 値=", args.CellValue1.ToString(),
        " 行=", args.RowIndex2, " 値=", args.CellValue2.ToString() );
end;
{ ----- } //選択処理
DGV1.cellclick += DGV1_cellclick; //セルをクリックした時
DGV1.cellvaluechanged += DGV1_cellvaluechanged; //値を変えた時
DGV1.columnheadermouseclick += DGV1_columnheadermouseclick; //列ヘッダーをクリックした時
DGV1.rowheadermouseclick += DGV1_rowheadermouseclick; //行ヘッダーをクリックした時
DGV1.selectionchanged += DGV1_selectionchanged; //選択を変えた時
DGV1.sortcompare += DGV1_sortcompare; //ソートした時
{ ----- }
```

アプリの作り方

プラットフォームヘルプのフォームクラスを見てアプリの作り方を簡単にまとめてみました
詳しくはプラットフォームヘルプ、フォームクラス内以下全てのヘルプを参照一読してみてください



フォーム例

フォームオブジェクトは2つのカテゴリー（コンテナ、コントロール）がある

コンテナ : フォーム、グループボックス、パネル等（コントロールのグループ化や表示に使用）

コントロール : ボタン、チェックボックス、コンボボックス、ラベル、リストビュー、ラジオボタン等（ユーザーが操作するパーツ）

主な作成手順

- ・コンテナ、コントロールの変数宣言（フォームの名前空間はelsystem.windows.forms）
- ・コンテナ及びコントロールを作成したオブジェクトに割り当てインスタンス初期化
- ・コントロールをコンテナに追加
- ・ロケーション設定、コンテナ内のコントロールの位置や大きさ等指定（左上が基準X,Y）
- ・イベントハンドラー追加（ボタンクリックや選択、入力等イベント発生時に呼び出すメソッドを追加）
- ・イベント発生処理記述（イベント発生時に呼び出されるメソッドの内容を記述）

あとはファイルの読み書き等、処理したい内容の記述でアプリ作りの流れはひとまず完了です
まあ本当の開発はテストやデバッグ作業ですけどね！

シンボルリンク

作成しようとしてるアプリにシンボルリンク機能をつけようと思い少し調べてみました

ウインドウタイプ3種類

マスター専用

シンボル/インターバルの変更情報をリンクグループ内の他のウインドウに提供、
リンクグループ内の他のウインドウで発生した変更には影響を受けない
(レーダースクリーン、トレードマネージャー)

スレーブ専用

リンクグループ内の他のウインドウからのシンボル/インターバルの変更を反映する
スレーブ専用ウインドウで発生した変更はリンクグループ内の他のウインドウに影響しない

マスター - スレーブ

リンクグループ内の他のウインドウのシンボル/インターバルに影響し、
リンクグループ内の他のウインドウで発生した変更を反映する場合がある
他のウインドウでシンボル/インターバルが変更されるとこのウインドウにも影響を与える

業績推移

レーダースクリーンのインジケータで入力項目に1期毎遡って入力して表示させたり
同じ銘柄名が何行も出たりとちとダサかったので公表はしてませんが
以前インジケータで作成した業績推移のアプリ版です、表示内容は株探とほぼ同じですw
トレステ内でサッと見れたり、実績データが徐々に溜まってくのでその分少し良いかなあと思ってます

アプリ1本目という事もあって分からない事が多く、インジケータの数倍理解と制御が大変でした(´д`;)
公式のアプリのソースも3本しかなくどれも長いのであまり参考にならず
あちこちに沼や谷や間があって、もう肝心な知りたいところだけヘルプのリンクが違うとか！
しかも2つも！イライラさせてくれるw
特にシンボルリンク、ファイルのロードの挙動と制御、データグリッドビューと行と列の記述と理解が大変でした
まだよく分かってない部分も多々あるので効率的ではない記述もあるかもしれませんが
これからアプリ作ってみたい人の参考になれば幸いです
アプリが作れると色々なデータを組み合わせでオリジナルなものが作れるので是非チャレンジを！
苦労した分出来た時にはちょっと感動しますw

現役時代の全盛期、社内の辞めた人が作った訳わからない長いプログラムで凄く悩んで寝ながらデバッグして翌朝その通りに作成したらプログラム完成した
事が1度だけあったのですがまさかの2度目がトレステのELで起こりましたw
寝勝手の極意！というか願っての極意と言うか睡眠時デバッグ症候群です
そんな悩んだプログラムじっくりご覧くださいw

注意事項

- ・インジケータ作成ではなくアプリ作成です
- ・プログラム内にも注記してありますが検証する時にイニシャライズイベントの設定を忘れないで下さい
忘れると真っ黒なフォームだけが現れますw
- ・現在の過去のファンダ実績データは6年分あるみたいですのでデフォルト表示7年にしました
- ・現状通期の実績のみです、四半期は未処理で今後の予定（未定です）
- ・連結/単独の入力項目いちいち入力するの面倒くさいと思ったので内部で判断するようにしました
普段自分がよく見る銘柄が正しく表示されるようにいくつか選択肢用意したのでコメントで制御するようにして下さい
デフォルトは件数の多い方で判断しています
- ・銘柄コードは空白でEnter押すと銘柄検索窓が開くようにしてあります
- ・表示年数は1-99年以外はエラーにしています
- ・万が一アプリがクラッシュしたりした場合はアプリをアクティブにして表示（V）再ロード（E）
又はCtrl + Rでアプリ再ロードになります
- ・トレステまでクラッシュしてしまった場合、私の場合は全ていいえでエラー状態を保存しないようにして終了させ
PCを再起動か、裏で動いているテレステ系のプロセスを全て終了させてトレステ再起動のどちらかをさせています

● プログラム

```
{-----}  
//宣言処理  
{-----} //クラス宣言  
using elsystem; //tsdataクラスによって使用される基本クラスと、その他一般的なelsystemクラス  
using platform; //主要プラットフォームおよび設定で使用されるクラス  
using elsystem.io; //入力/出力システム例外の処理に使用される基本クラス  
using tsdata.marketdata; //価格クォート、市場レベル、ファンダメンタル値などの市場データへのアクセスに使用されるクラス  
using elsystem.collections; //各種のコレクションオブジェクトの作成に使用される基本クラス  
using tsdata.common; //他のtsdata名前空間によって使用されるクラス  
using elsystem.windows.forms; //フォームのコントロールおよびコンテナの作成に使用されるクラス  
using elsystem.drawing; //フォームコントロールやドローイングオブジェクトの色及びフォントを記述する為に使用されるクラス
```

業績推移 (2)

```
{ ----- } //変数宣言 (単数は0:省略可は省略、複数は1から連番)
variables:
    Form MainForm( null ),                //フォーム      Formクラス
    Panel Panel0( null ),                 //パネル       Panelクラス
    Label Label1( null ), Label Label2( null ), Label Label3( null ), //ラベル      Labelクラス
    TextBox TextBox1( null ), TextBox TextBox2( null ), //テキストボックス TextBoxクラス
    RadioButton RdBtn1( null ), RadioButton RdBtn2( null ), //ラジオボタン RadioButtonクラス
    DataGridView DGV0( null ),           //表           DataGridViewクラス
    SymbolLinking SLink( null ),         //シンボルリンク SymbolLinkingクラス
    SymbolContext SLContext ( null ),     //シンボルリンク SymbolContextクラス
    SymbolLookupDialog symbolLookup( null ), //銘柄検索    CommonDialogクラス
    SymbolAttributesProvider SAP( null ), //コンポーネント SymbolAttributesProviderクラス
    FundamentalQuotesProvider FQP( null ), //コンポーネント FundamentalQuotesProviderクラス
    Dictionary DataDict( null ),         //辞書        Dictionaryクラス (コレクション) キー/値のペアのコレクション
    int PeriodsAgo( 0 ),                  //遡り        : 0=最新 大きい程過去
    int MonthsReported( 12 ),//期間: 3, 6, 12 (数字の期間で纏められてる、3: 四半期、6: 半期、12: 通期) 1、3四半期は同じ3
    int ConsolidationLevel( 1 ),          //連結/単独   : 1=連結 1≠単独
    int AccountingStandard( 0 ),         //会計基準    : 1= JSTD 2=SEC 3=IFRS
    string ConsLevel( "Consolidated" ), //連結/単独
    string AcctStdInputString( "" ),     //会計基準
    intrabarpersist bool OkSAP( false ), //SAP銘柄コード確認用
    intrabarpersist bool OkFQP( false ), //FQP銘柄コード確認用
    intrabarpersist bool OkYear( true ), //年数確認用
    intrabarpersist bool CodeOnce( true ), //銘柄変更毎1回用
    intrabarpersist string KpCode( "" ), //銘柄変更確認用
    intrabarpersist string KpYear( "" ), //年数変更確認用
    double FundFieldValue( 0 ),          //値
    DateTime Happyo( null ),             //決算発表日
    int PeriodsAgoPlus( 0 ),             //前期比用
    int RowCount( 0 );                   //行数
{ ----- } //定数宣言
constants:
    string MonthsReportedKey( "MonthsReported" ), //期間
    string ConsolidatedKey( "ConsolidationLevel" ), //連結/単独
    string AccountingKey( "AccountingStandard" ), //会計基準
    string SALESActualFieldName( "CR_SALES" ), //実売上
    string OPActualFieldName( "CR_Op" ), //実営利
    string RPAActualFieldName( "CR_RP" ), //実経常
    string NPAActualFieldName( "CR_NP" ), //実純利
    string EPSActualFieldName( "CR_EPS" ), //1株益
    string DPSActualFieldName( "CR_DPS" ), //1株配
    string SALESForecastFieldName( "CE_SALES" ), //予売上
    string OPForecastFieldName( "CE_Op" ), //予営利
    string RPFforecastFieldName( "CE_RP" ), //予経常
    string NPFforecastFieldName( "CE_NP" ), //予純利
    string EPSForecastFieldName( "CE_EPS" ), //予1株益
    string DPSForecastFieldName( "CE_DPS" ), //予1株配
    string ConsolidatedValue( "Consolidated" ), //連結
    string NonConsolidatedValue( "NonConsolidated" ), //単独
    string ValuesAdder( "_Values" ), //値
    string IFRSString( "IFRS" ), //会計基準(IFRS)
    string SECString( "SEC" ), //会計基準(SEC)
    string JSTDString( "JSTD" ); //会計基準(JSTD)
```

業績推移 (3)

```
{ ----- }
//初期処理 ※ 検証する時に、プロパティでイニシャライズイベントに設定
{ ----- }
method void AnalysisTechnique_Initialized( elsystem.Object sender, elsystem.InitializedEventArgs args )
begin
//各種パーツ初期設定・表示
    MainForm_Column(); //メインフォーム
    SymbolLink_Column(); //シンボルリンク
    DGV_Column(); //表
    Header_Column(); //列見出し
    Label_Column(); //ラベル (文字、数値) ※ パネルの前にはないとパネルの下にラベルが隠れる
    TextBox_Column(); //テキストボックス ※ パネルの前にはないとパネルの下にテキストが隠れる
    RadioButton_Column(); //ラジオボタン ※ パネルの前にはないとパネルの下にラジオボタンが隠れる
    Panel_Column(); //パネル ※ 表のあとパネルにはないとパネルの下に表が隠れる
    MainForm.Show(); //メインフォームを表示
end; { method AnalysisTechnique_Initialized }
{ ----- }
//各種パーツ初期設定
{ ----- } //メインフォーム初期設定
method void MainForm_Column()
begin
    MainForm = Form.Create(); //フォームの新規インスタンスを初期化
    MainForm.Dock = DockStyle.Fill; //フォームのコントロールがドッキングされる位置および方法を指定
end; { Form_Column }
{ ----- } //シンボルリンク初期設定
method void SymbolLink_Column()
begin
    SLink = SymbolLinking.Create();
    SLContext = SymbolContext.Create();
    SLink.GetContext += GetContext_SymbolLink; //シンボルリンクアイコン選択した時
    SLink.SetContext += SetContext_SymbolLink; //シンボルリンク銘柄変更した時
end; { SymbolLink_Column }
{ ----- } //表初期設定
method void DGV_Column()
begin
    DGV0 = DataGridView.Create(); //DataGridViewクラス、DGV0の新規インスタンスを初期化
    MainForm.AddControl( DGV0 ); //コントロールをMainFormに追加
    DGV0.Dock = DockStyle.fill; //DGV0のコントロールがドッキングされる位置および方法を指定
    DGV0.AllowUserToAddRows = false; //ユーザーがグリッド行を追加できる場合はtrue/false
    DGV0.BackColor = color.Black; //ウィンドウ背景色
    DGV0.DefaultCellStyle.BackColor = color.Black; //セル背景色
    DGV0.DefaultCellStyle.ForeColor = color.OliveDrab; //文字色
    DGV0.GridColor = color.DimGray; //グリッド色
    DGV0.ReadOnly = true; //ユーザーが列のセルを編集できる場合はfalse/true
    DGV0.ColumnHeadersDefaultCellStyle.Alignment//ヘッダーのデフォルトの整列スタイルを取得または設定( Left:0 Center:1 Right:2 )
    = DataGridViewContentAlignment.MiddleCenter;
    DGV0.DefaultCellStyle.Format = "###,###,###,###,##0";//フォーマットカンマ付き数字スタイルにしたいがやり方が分からない!
    DGV0.DefaultCellStyle.Alignment = ContentAlignment.MiddleRight;//整列スタイルを取得または設定( Left:0 Center:1 Right:2 )
end; { DGV_Column }
```

業績推移 (4)

```
{ ----- } //列見出し初期設定
method void Header_Column()
begin
    DGV0.RowHeadersVisible = false;           //列のヘッダーが表示される場合true/false
    DGV0.RowHeadersWidth = 20;                //列のヘッダーの幅
    DGV0.ColumnHeadersHeight = 32;           //列のヘッダーの高さ
    DGV0.ColumnHeadersHeightSizeMode = 0;     //列のヘッダーの高さ調整 (0:可1:不可2:AutoSize)
    DGV0.Columns.Add( "No" );                 //列見出し0
    DGV0.Columns.Add( "売上" );               //列見出し1
    DGV0.Columns.Add( "営利" );               //列見出し2
    DGV0.Columns.Add( "経常" );               //列見出し3
    DGV0.Columns.Add( "純利" );               //列見出し4
    DGV0.Columns.Add( "1株益" );              //列見出し5
    DGV0.Columns.Add( "1株配" );              //列見出し6
    DGV0.Columns.Add( "利益率%" );            //列見出し7
    DGV0.Columns.Add( "営業利益前期比" );     //列見出し8
    DGV0.Columns.Add( "営業利益前期比%" );    //列見出し9
    DGV0.Columns.Add( "決算日" );             //列見出し10
    DGV0.Columns[0].Width = 20;                //列の幅 (ピクセル単位) を取得または設定
    DGV0.Columns[0].DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter; //テキストのデフォルトの整列スタイルを取得または設定 //DataGridViewContentAlignment列挙参照
    DGV0.Columns[1].Width = 70;
    DGV0.Columns[2].Width = 70;
    DGV0.Columns[3].Width = 70;
    DGV0.Columns[4].Width = 70;
    DGV0.Columns[5].Width = 45;
    DGV0.Columns[6].Width = 45;
    DGV0.Columns[7].Width = 50;
    DGV0.Columns[8].Width = 70;
    DGV0.Columns[9].Width = 70;
    DGV0.Columns[10].Width = 70;
    DGV0.Columns[5].DefaultCellStyle.Format = "###,###,###,###,##0.0"; //フォーマットカンマ付き数字スタイルにしたい!
    DGV0.Columns[6].DefaultCellStyle.Format = "###,###,###,###,##0.0";
    DGV0.Columns[7].DefaultCellStyle.Format = "###,##0.0%"; //フォーマット%スタイルにしたい!
    DGV0.Columns[9].DefaultCellStyle.Format = "###,##0.0%";
end; { Header_Column }

{ ----- } //パネル初期設定
method void Panel_Column()
begin
    Panel0 = Panel.Create( 0, 30 );           //パネルの新規インスタンスを初期化 (幅、高さ)
    Panel0.Dock = DockStyle.top;              //パネルのコントロールがドッキングされる位置および方法を指定
    Panel0.BackColor = color.Gainsboro;       //背景色
    MainForm.AddControl( Panel0 );            //コントロールをMainFormに追加
end; { Panel_Column }
```

業績推移 (5)

```
{ ----- } //ラベル初期設定
method void Label_Column()
begin
//ラベル1 (銘柄コード (テキスト))
    Label1 = Label.create( "銘柄コード", 60, 15 );           //ラベル (テキスト、幅、高さ)
    Label1.Location( 25, 8 );                               //左上基準位置 ( X,Y )
    Label1.BackColor = color.Gainsboro;                    //背景色
    MainForm.AddControl( Label1 );                         //コントロールをMainFormに追加
//ラベル2 (銘柄名)
    Label2 = Label.Create( "", 300, 15 );                  //ラベル (枠のみ、コード入力で銘柄名表示)
    Label2.Location( 135, 8 );                             //左上基準位置 ( X,Y )
    Label2.BackColor = color.Gainsboro;                    //背景色
    MainForm.AddControl( Label2 );                         //コントロールをMainFormに追加
//ラベル3 (年 (テキスト))
    Label3 = Label.Create( "年", 15, 15 );                //ラベル (テキスト、幅、高さ)
    Label3.Location( 630, 8 );                             //左上基準位置 ( X,Y )
    Label3.BackColor = color.Gainsboro;                    //背景色
    MainForm.AddControl( Label3 );                         //コントロールをMainFormに追加
end; { Label_Column }
{ ----- } //テキストボックス初期設定
method void TextBox_Column()
begin
//テキストボックス1 (銘柄コード入力用)
    TextBox1 = Textbox.create( "", 40, 20 );              //テキストボックス (テキスト、幅、高さ)
    TextBox1.Location( 90, 5 );                           //左上基準位置 ( X,Y )
    MainForm.AddControl( TextBox1 );                       //コントロールをMainFormに追加
    TextBox1.KeyUp += TextBox_KeyUp;                       //テキストボックスに入力した時
//テキストボックス2 (表字年数入力用)
    TextBox2 = Textbox.create( "7", 20, 20 );             //テキストボックス (テキスト、幅、高さ)
    TextBox2.Location( 610, 5 );                           //左上基準位置 ( X,Y )
    MainForm.AddControl( TextBox2 );                       //コントロールをMainFormに追加
    TextBox2.KeyUp += TextBox_KeyUp;                       //テキストボックスに入力した時
end; { TextBox_Column }
{ ----- } //ラジオボタン初期設定
method void RadioButton_Column()
begin
//ラジオボタン1 (通期)
    RdBtn1 = RadioButton.create( " 通 期", 60, 20 );      //ラジオボタン (テキスト、幅、高さ)
    RdBtn1.Location( 460, 5 );                             //左上基準位置 ( X,Y )
    RdBtn1.BackColor = color.Gainsboro;                    //背景色
    RdBtn1.Checked = true;                                  //ラジオボタン (初期値ON true)
    MainForm.AddControl( RdBtn1 );                         //コントロールをMainFormに追加
    RdBtn1.Click += RdBtn1_Click;                          //ラジオボタン1 をクリックした時
//ラジオボタン2 (四半期)
    RdBtn2 = RadioButton.create( " 四半期", 65, 20 );    //ラジオボタン (テキスト、幅、高さ)
    RdBtn2.Location( 530, 5 );                             //左上基準位置 ( X,Y )
    RdBtn2.BackColor = color.Gainsboro;                    //背景色
    RdBtn2.Checked = false;                                 //ラジオボタン (初期値OFF false)
    MainForm.AddControl( RdBtn2 );                         //コントロールをMainFormに追加
    RdBtn2.Click += RdBtn2_Click;                          //ラジオボタン2 をクリックした時
end; { RadioButton_Column }
```

業績推移 (6)

```
{ ----- }
//イベント発生処理
{ ----- } //シンボルリンクアイコン選択した時
method void GetContext_SymbolLink( elsystem.Object sender, SymbolLinkingEventArgs args )
begin
    args.Recalculate = false; //アプリ再計算true/false
    TextBox1.Text = args.Symbol.Substring(0,4);
    if TextBox1.text.Length >= 4 and OkYear = true then CheckCode(); //銘柄コードチェック
end; { SetContext_SymbolLink }
{ ----- } //シンボルリンク銘柄変更した時
method void SetContext_SymbolLink( elsystem.Object sender, SymbolLinkingEventArgs args )
begin
    args.Recalculate = false; //アプリ再計算true/false
    TextBox1.Text = args.Symbol.Substring(0,4);
    if TextBox1.text.Length >= 4 and OkYear = true then CheckCode(); //銘柄コードチェック
end; { SetContext_SymbolLink }
{ ----- } //テキストボックスに入力した時
method void TextBox_KeyUp( elsystem.Object sender, elsystem.windows.forms.KeyEventArgs args )
begin
    if args.KeyCode.ToString() = "Return" then begin //予約語ReturnじゃなくEnterキー、args.KeyCode = Keys.Returnでも可
//表示年数チェック
        if TextBox2.text <= "0"
        or TextBox2.text > "99"
        or TextBox2.text = ""
        or TextBox2.text.Length >= 3 then OkYear = false else OkYear = true; //年数確認用
        if OkYear = false then Label2.Text = "表示年数が正しくありません"; //銘柄名
//銘柄コード長0の時、銘柄検索 //銘柄コードAlt+(B)に初期設定したいがフィールド名が分からない！
        if OkYear = true and TextBox1.text.Length = 0 then begin
            SymbolLookup = SymbolLookupDialog.Create();
            SymbolLookup.StatusChanged += SymbolLookupStatus_DialogChanged;
            SymbolLookup.Show();
        end;
//年数エラーなし、銘柄コード長4以上の時、銘柄コードチェック
        if OkYear = true and TextBox1.text.Length >= 4 then CheckCode();
    end;
end; { TextBox_KeyUp }
{ ----- } //ラジオボタン1をクリックした時
method void RdBtn1_Click( elsystem.Object sender, elsystem.EventArgs args )
begin
    Clearprintlog(); print( "通期をクリック！期間MonthsReportedに12をセット" );
    MonthsReported = 12; //期間に12をセット
end; { RdBtn1_Click }
{ ----- } //ラジオボタン2をクリックした時
method void RdBtn2_Click( elsystem.Object sender, elsystem.EventArgs args )
begin
    Clearprintlog(); print( "四半期をクリック、四半期処理はまだ未処理！" );
end; { RdBtn2_Click }
{ ----- } //銘柄検索OKした時
method void SymbolLookupStatus_DialogChanged( Object sender, DialogStatusChangedEventArgs args )
begin
    if symbolLookup.Status = DialogResult.OK then TextBox1.Text = symbolLookup.Symbol.Substring(0,4);
    if TextBox1.text.Length >= 4 and OkYear = true then CheckCode(); //銘柄コードチェック
end; { SymbolLookupStatus_DialogChanged }
```

業績推移 (7)

```
{ ----- } //SAP変更した時
method void SAP_Updated( Object sender, SymbolAttributesUpdatedEventArgs args )
begin
    switch ( SAP.State ) begin
        case 3:{ failed }
            OkSAP = false; //銘柄コード確認NG
            Label2.Text = "銘柄コードが正しくありません"; //銘柄名
        case 2:{ loaded }
            OkSAP = true;
            Label2.Text = SAP.Description.ToString(); //銘柄名
            CreateFQP(); //FQP初期設定
        case 1:{ loading }
            OkSAP = false;
            Label2.Text = "銘柄データloading もう一度Enterを押して下さい"; //銘柄名
        case 0:{ Unloaded }
            OkSAP = false;
    end;
end; { SAPFQP_StateChanged }
{ ----- } //FQP変更した時
method void FQP_Updated( Object sender, FundamentalQuoteUpdatedEventArgs args )
begin
    switch ( FQP.State ) begin
        case 3:{ failed }
            OkFQP = false; //銘柄コード確認NG
            Label2.Text = "銘柄データがありません"; //銘柄名
        case 2:{ loaded }
            OkFQP = true;
            PeriodsAgo = int.Parse( KpYear ); //遡り数
            RowCount = 0; //行数
            for PeriodsAgo = PeriodsAgo - 1 downto 0 begin //最新まで逆読み
                FQPDataSet(); //ファンダメンタルデータセット・実績表示
                RowCount = RowCount + 1; //DGV行数+1
            end;
            DGV0.Sort( DGV0.Columns[0], 1 ); //列のソートモードを取得または設定 (0:昇順 1:降順)
            FQPDataSet2(); //ファンダメンタルデータセット・予想表示
        case 1:{ loading }
            OkFQP = false;
        case 0:{ Unloaded }
            OkFQP = false;
    end;
end; { SAPFQP_StateChanged }
```

業績推移 (8)

```
{ ----- }
//その他サブルーチン処理
{ ----- } //銘柄コードチェック、銘柄名セット (ラベル2)
method void CheckCode()
begin
//銘柄、表示年数変更時クリア
if TextBox1.text = "" or TextBox1.text <> KpCode
or TextBox2.text = "" or TextBox2.text <> KpYear then begin
    Label2.Text = ""; //銘柄名クリア
    DGV0.Rows.clear(); //行クリア
    DataDict = new Dictionary(); //辞書初期化
    OkSAP = false; //SAP銘柄コード確認用
    OkFQP = false; //FQP銘柄コード確認用
    KpCode = TextBox1.text; //銘柄変更確認用
    KpYear = TextBox2.text; //年数変更確認用
    CodeOnce = true; //銘柄変更毎1回用
end;
CreateSAP(); //SAP初期設定
if OkSAP = false then Label2.Text = "銘柄コードが正しくありません"; //銘柄名
end; { CheckCode method }
{ ----- } //SymbolAttributesProvider初期設定
method void CreateSAP()
begin
    SAP = SymbolAttributesProvider.Create(); //プロバイダーオブジェクトを作成
    SAP.Symbol = TextBox1.text; //シンボル
    if OkSAP = false then SAP.Updated += SAP_Updated; //変更イベント時 (銘柄確認まだの時のみ)
    SAP.Load = true; //プロバイダー接続
end; { CreateSAP }
{ ----- } //FundamentalQuotesProvider初期設定
method void CreateFQP()
begin
    FQP = FundamentalQuotesProvider.Create(); //プロバイダーオブジェクトを作成
    FQP.Symbol = TextBox1.text; //シンボル
    FQP.Fields += SALESActualFieldName; //実売上
    FQP.Fields += OPAActualFieldName; //実営利
    FQP.Fields += RPAActualFieldName; //実経常
    FQP.Fields += NPAActualFieldName; //実純利
    FQP.Fields += EPSActualFieldName; //1株益
    FQP.Fields += DPSActualFieldName; //1株配
    FQP.Fields += SALESForecastFieldName; //予売上
    FQP.Fields += OPForecastFieldName; //予営利
    FQP.Fields += RPForecastFieldName; //予経常
    FQP.Fields += NPForecastFieldName; //予純利
    FQP.Fields += EPSForecastFieldName; //予1株益
    FQP.Fields += DPSForecastFieldName; //予1株配
    if OkFQP = false then FQP.Updated += FQP_Updated; //変更イベント時 (銘柄確認まだの時のみ)
    FQP.Load = true; //プロバイダー接続
end; { CreateFQP }
```

業績推移 (9)

```

{ ----- }
//ファンダメンタルデータセット処理
{ ----- } //データ存在チェック、セット
method bool GetQuoteAsOfDate( string FieldName, DateTime tempdt, int PeriodsAgo, out double QuoteVal )
variables:
    int Counter,
    bool QuoteFound, bool SetVariable,
    DateTime VectDateTime,
    Vector QuoteDateTimeVector, Vector QuoteValuesVector;
begin
    if FieldName = "" or tempdt = null or DataDict = null or PeriodsAgo < 0 then return false;
    QuoteDateTimeVector = DataDict[FieldName] astype Vector;
    QuoteValuesVector = DataDict[FieldName + ValuesAdder] astype Vector;
    QuoteFound = false;
    Happyo = DateTime.Create( 1900, 1, 1 );
    for Counter = QuoteDateTimeVector.Count - 1 downto 0 begin
        SetVariable = false;
        VectDateTime = QuoteDateTimeVector[Counter] astype DateTime;
        if tempdt >= VectDateTime then begin { 現在日時以下ならデータセット }
            SetVariable = true;
        end else begin
            break;           { for downto 終了 return へ }
        end;
        if SetVariable then begin
            if Counter + PeriodsAgo <= QuoteDateTimeVector.Count - 1 then begin
                QuoteVal = QuoteValuesVector[Counter + PeriodsAgo] astype double; //値
                Happyo = QuoteDateTimeVector[Counter + PeriodsAgo] astype DateTime; //決算発表日
                QuoteFound = true;
            end;
        end;
    end;
    return QuoteFound; { メソッドの実行を終了し、メソッドの呼び出し元に制御を返す、リターンキーワードQuoteFoundはbool型なのでtrue/falseどちらかを返す }
end;{ GetQuoteAsOfDate }

```

業績推移アプリ完成図

No	売上	営利	経常	純利	1株益	1株配	利益率%	営業利益 前期比	営業利益 前期比%	決算日
6										
5	18,583,653	355,627	432,873	283,559	90.2	50.0	1.9%			2012/05/09
4	22,064,192	1,320,888	1,403,649	962,163	303.8	90.0	6.0%	965,261	271.4%	2013/05/08
3	25,691,911	2,292,112	2,441,080	1,823,119	575.3	165.0	8.9%	971,224	73.5%	2014/05/08
2	27,234,521	2,750,564	2,892,828	2,173,338	688.0	200.0	10.1%	458,452	20.0%	2015/05/08
1	28,403,118	2,853,971	2,983,381	2,312,694	741.4	210.0	10.0%	103,407	3.8%	2016/05/11
0	27,597,193	1,994,372	2,193,825	1,831,109	605.5	210.0	7.2%	-859,599	-30.1%	2017/05/10
予	28,500,000	2,000,000	2,250,000	1,950,000	657.1		7.0%	5,628	0.3%	

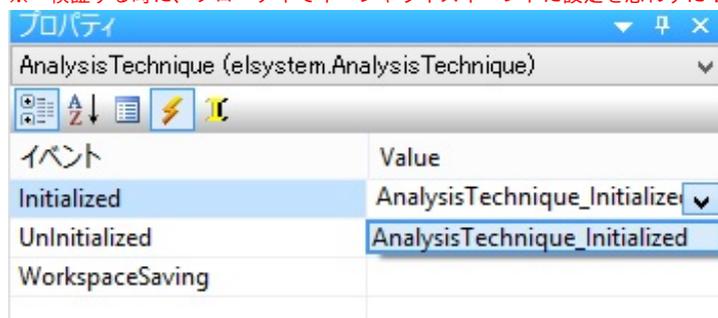
業績推移 (10)

```
{ ----- } //Vectorクラスデータ要素のコレクションを作成
method void LoadFundDataVectorsWithFilters( string ifundField )
variables:
    int Counter,
    Vector dateVect, Vector valuesVect,
    FundamentalQuote fq,
    string fqAcctStd, string LastAcctStd,
    bool LoadThisQuote, bool ReplaceLastValue, bool NumMonthsOkay, bool ConsLevelOkay,
    DateTime LastDateTime;
begin
    if DataDict = null then DataDict = new Dictionary();
    dateVect = new Vector();
    valuesVect = new Vector();
    DataDict.Add( ifundField, dateVect astype Vector );
    DataDict.Add( ifundField + ValuesAdder, valuesVect astype Vector );
    fq = FQP[ifundField];
    LastDateTime = DateTime.Create( 1900, 1, 1 );
    for Counter = 0 to fq.Count - 1 begin
        LoadThisQuote = false;
        ReplaceLastValue = false;
        fqAcctStd = fq.ExtendedProperties[Counter][AccountingKey].StringValue;
        NumMonthsOkay = fq.ExtendedProperties[Counter][MonthsReportedKey].IntegerValue = MonthsReported;
        ConsLevelOkay = fq.ExtendedProperties[Counter][ConsolidatedKey].StringValue = ConsLevel;
        if NumMonthsOkay and ConsLevelOkay and ( AccountingStandard = 0 or fqAcctStd = AcctStdInputString ) then begin
            if fq.PostDate[Counter] <> LastDateTime then begin { new date }
                LoadThisQuote = true;
            end else begin { same date as last quote loaded }
                switch ( fqAcctStd ) begin
                    case IFRSString:ReplaceLastValue = true;
                    case SECString: if LastAcctStd <> IFRSString then ReplaceLastValue = true;
                    case JSTDString:if LastAcctStd <> IFRSString and LastAcctStd <> SECString then ReplaceLastValue = true;
                end;
            end;
        end;
        if LoadThisQuote then begin
            dateVect.push_back( fq.PostDate[Counter] astype DateTime ); //コレクションの最後に新規要素 (日付) を追加
            valuesVect.push_back( fq.DoubleValue[Counter] astype double ); //コレクションの最後に新規要素 (値) を追加
            LastAcctStd = fqAcctStd;
            LastDateTime = fq.PostDate[Counter];
        end else if ReplaceLastValue then begin
            valuesVect[Counter-1] = fq.DoubleValue[Counter] astype double;
            LastAcctStd = fqAcctStd;
        end;
    end;
end;{ LoadFundDataVectorsWithFilters method }
```

業績推移 (11)

```
{ ----- } //連結/単独セット
method string GetConsLevel()
variables:
    int Counter, int ConsCount, int NonConsCount,
    FundamentalQuote fq,
    string ConsLevel;
begin
    fq = FQP[SALESActualFieldName]; //売上実績データで件数確認
    ConsCount = 0; NonConsCount = 0;
    for Counter = 0 to fq.Count - 1 begin
        if fq.ExtendedProperties[Counter][ConsolidatedKey].StringValue = ConsolidatedValue then ConsCount = ConsCount + 1 else NonConsCount =
NonConsCount + 1;
    { ※ どれか選択 }
        //最初のデータを採用する場合
    // if Counter = 0 then
    // if fq.ExtendedProperties[Counter][ConsolidatedKey].StringValue = ConsolidatedValue then ConsLevel = ConsolidatedValue else ConsLevel =
NonConsolidatedValue
    // else
        //最後のデータを採用する場合
    // if fq.ExtendedProperties[Counter][ConsolidatedKey].StringValue = ConsolidatedValue then ConsLevel = ConsolidatedValue else ConsLevel =
NonConsolidatedValue;
    end;
    //全件読んでConsol、NonConsol、件数の多い方採用する場合（同数は連結）
    if ConsCount >= NonConsCount then ConsLevel = ConsolidatedValue else ConsLevel = NonConsolidatedValue;
    return ConsLevel;
end;{ GetConsol method }
{ ----- } //会計基準セット
method string GetAcctStdString()
variables: string AcctStdString;
begin
    switch ( AccountingStandard ) begin
        case 1: AcctStdString = JSTDString;
        case 2: AcctStdString = SECString;
        case 3: AcctStdString = IFRSString;
        default:AcctStdString = "";
    end;
    return AcctStdString;
end;{ GetAcctStdString method }
```

※ 検証する時に、プロパティでイニシャライズイベントに設定を忘れずに！



業績推移 (12)

```
{ ----- } //ファンダメンタルデータセット・実績表示
method void FQPDataSet()
begin
//行追加、設定
    DGV0.Rows.Add( PeriodsAgo ); //行追加、遡りNoセット
    DGV0.Rows[RowCount].Height = 18; //行の高さ (ピクセル単位) を取得または設定
    DGV0.Rows[RowCount].Resizable = 2; //行のサイズ変更を取得または設定 ( 0:未設定 1 : true 2 : False )
//連結/単独セット
    if CodeOnce then ConsLevel = GetConsLevel();
//会計基準セット
    if CodeOnce then AcctStdInputString = GetAcctStdString();
//実売上( Value1 )
    if FQP.HasQuoteData( SALESActualFieldName ) and ( FQP[SALESActualFieldName].Type = DataType.doubleval //指定名を持つクォートに値が含まれる場合は真
or FQP[SALESActualFieldName].Type = DataType.integral ) then begin
        Condition1 = true;
        if CodeOnce then if FQP[SALESActualFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
SALESActualFieldName ); //指定名を持つクォートがプロバイダーに存在する場合は真
    end;
    if Condition1 then begin
        if GetQuoteAsOfDate( SALESActualFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value1 = FundFieldValue;
        DGV0.Rows[RowCount].Cells[1].Value = ( Value1 );
    end;
//実営利( Value2 )
    if FQP.HasQuoteData( OPActualFieldName ) and ( FQP[OPActualFieldName].Type = DataType.doubleval
or FQP[OPActualFieldName].Type = DataType.integral ) then begin
        Condition2 = true;
        if CodeOnce then if FQP[OPActualFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
OPActualFieldName );
    end;
    if Condition2 then begin
        if GetQuoteAsOfDate( OPActualFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value2 = FundFieldValue;
        DGV0.Rows[RowCount].Cells[2].Value = ( Value2 );
    end;
//実経常( Value3 )
    if FQP.HasQuoteData( RPAcualFieldName ) and ( FQP[RPAcualFieldName].Type = DataType.doubleval
or FQP[RPAcualFieldName].Type = DataType.integral ) then begin
        Condition3 = true;
        if CodeOnce then if FQP[RPAcualFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
RPAcualFieldName );
    end;
    if Condition3 then begin
        if GetQuoteAsOfDate( RPAcualFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value3 = FundFieldValue;
        DGV0.Rows[RowCount].Cells[3].Value = ( Value3 );
    end;
//実純利( Value4 )
    if FQP.HasQuoteData( NPAcualFieldName ) and ( FQP[NPAcualFieldName].Type = DataType.doubleval
or FQP[NPAcualFieldName].Type = DataType.integral ) then begin
        Condition4 = true;
        if CodeOnce then if FQP[NPAcualFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
NPAcualFieldName );
    end;
    if Condition4 then begin
        if GetQuoteAsOfDate( NPAcualFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value4 = FundFieldValue;
        DGV0.Rows[RowCount].Cells[4].Value = ( Value4 );
    end;
end;
```

業績推移 (13)

```
//1株益( Value5 )
if FQP.HasQuoteData( EPSActualFieldName ) and ( FQP[EPSActualFieldName].Type = DataType.doubleval
or FQP[EPSActualFieldName].Type = DataType.integral ) then begin
    Condition5 = true;
    if CodeOnce then if FQP[EPSActualFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
EPSActualFieldName );
end;
if Condition5 then begin
    if GetQuoteAsOfDate( EPSActualFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value5 = FundFieldValue;
    DGV0.Rows[RowCount].Cells[5].Value = ( Value5 );
end;
//1株配( Value6 )
if FQP.HasQuoteData( DPSActualFieldName ) and ( FQP[DPSActualFieldName].Type = DataType.doubleval
or FQP[DPSActualFieldName].Type = DataType.integral ) then begin
    Condition6 = true;
    if CodeOnce then if FQP[DPSActualFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
DPSActualFieldName );
end;
if Condition6 then begin
    if GetQuoteAsOfDate( DPSActualFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value6 = FundFieldValue;
    DGV0.Rows[RowCount].Cells[6].Value = ( Value6 );
end;
//利益率%( Value7 )
if Value1 <> 0 then Value7 = ( Value2 / Value1 * 100 );
DGV0.Rows[RowCount].Cells[7].Value = ( Value7 );
//決算発表日
DGV0.Rows[RowCount].Cells[10].Value = ( Happyo.ToString() );
if Happyo.toString() = ( "1900/01/01" ) then DGV0.Rows[RowCount].Cells[10].ForeColor = color.Black;//背景同色
//前期比 ( Value8 )
PeriodsAgoPlus = PeriodsAgo + 1;
if GetQuoteAsOfDate( OPAActualFieldName, BarDateTime, PeriodsAgoPlus, FundFieldValue ) then begin
    if Condition2 then Value8 = Value2 - FundFieldValue;
end;
DGV0.Rows[RowCount].Cells[8].Value = ( Value8 );
//前期比%( Value9 )
if Value8 <> 0 then Value9 = ( ( Value2 - FundFieldValue ) / FundFieldValue * 100 );
DGV0.Rows[RowCount].Cells[9].Value = ( Value9 );
//銘柄変更毎 1 回用
CodeOnce = false;
//色変更
ColorSet();
end;{ FQPDataSet method }
```

業績推移 (14)

```
{ ----- } //ファンダメンタルデータセット2・予想表示
method void FQPDataSet2()
begin
//行追加、設定
    DGV0.Rows.Add( "予" ); DGV0.Rows[RowCount].Height = 18; DGV0.Rows[RowCount].Resizable = 2; PeriodsAgo = 0;
//予売上( Value1 )
    if FQP.HasQuoteData( SALESForecastFieldName ) and ( FQP[SALESForecastFieldName].Type = DataType.doubleval
or FQP[SALESForecastFieldName].Type = DataType.integral ) then begin
        Condition1 = true;
        if FQP[SALESForecastFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
SALESForecastFieldName );
        end;
    if Condition1 then begin
        if GetQuoteAsOfDate( SALESForecastFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value1 = FundFieldValue;
        DGV0.Rows[RowCount].Cells[1].Value = ( Value1 );
        end;
//予営利( Value2 )
    if FQP.HasQuoteData( OPForecastFieldName ) and ( FQP[OPForecastFieldName].Type = DataType.doubleval
or FQP[OPForecastFieldName].Type = DataType.integral ) then begin
        Condition2 = true;
        if FQP[OPForecastFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
OPForecastFieldName );
        end;
    if Condition2 then begin
        if GetQuoteAsOfDate( OPForecastFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value2 = FundFieldValue;
        DGV0.Rows[RowCount].Cells[2].Value = ( Value2 );
        end;
//予経常( Value3 )
    if FQP.HasQuoteData( RPFforecastFieldName ) and ( FQP[RPFforecastFieldName].Type = DataType.doubleval
or FQP[RPFforecastFieldName].Type = DataType.integral ) then begin
        Condition3 = true;
        if FQP[RPFforecastFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
RPFforecastFieldName );
        end;
    if Condition3 then begin
        if GetQuoteAsOfDate( RPFforecastFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value3 = FundFieldValue;
        DGV0.Rows[RowCount].Cells[3].Value = ( Value3 );
        end;
//予純利( Value4 )
    if FQP.HasQuoteData( NPForecastFieldName ) and ( FQP[NPForecastFieldName].Type = DataType.doubleval
or FQP[NPForecastFieldName].Type = DataType.integral ) then begin
        Condition4 = true;
        if FQP[NPForecastFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
NPForecastFieldName );
        end;
    if Condition4 then begin
        if GetQuoteAsOfDate( NPForecastFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value4 = FundFieldValue;
        DGV0.Rows[RowCount].Cells[4].Value = ( Value4 );
        end;
//予1株益( Value5 )
    if FQP.HasQuoteData( EPSForecastFieldName ) and ( FQP[EPSForecastFieldName].Type = DataType.doubleval
or FQP[EPSForecastFieldName].Type = DataType.integral ) then begin
        Condition5 = true; if FQP[EPSForecastFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then
LoadFundDataVectorsWithFilters( EPSForecastFieldName );
        end;
end;
```

業績推移 (15)

```
if Condition5 then begin
    if GetQuoteAsOfDate( EPSForecastFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value5 = FundFieldValue;
    DGV0.Rows[RowCount].Cells[5].Value = ( Value5 );
end;
//予1株配( Value6 )
if FQP.HasQuoteData( DPSForecastFieldName ) and ( FQP[DPSForecastFieldName].Type = DataType.doubleval
or FQP[DPSForecastFieldName].Type = DataType.integerval ) then begin
    Condition6 = true;
    if FQP[DPSForecastFieldName].ExtendedProperties[0].Contains( MonthsReportedKey ) then LoadFundDataVectorsWithFilters(
DPSForecastFieldName );
end;
if Condition6 then begin
    if GetQuoteAsOfDate( DPSForecastFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then Value6 = FundFieldValue;
    DGV0.Rows[RowCount].Cells[6].Value = ( Value6 );
end;
//予利益率%( Value7 )
if Value1 <> 0 then Value7 = ( Value2 / Value1 * 100 );
DGV0.Rows[RowCount].Cells[7].Value = ( Value7 );
//前期比 ( Value8 )
if GetQuoteAsOfDate( OPAActualFieldName, BarDateTime, PeriodsAgo, FundFieldValue ) then begin
    if Condition2 then Value8 = Value2 - FundFieldValue;
end;
DGV0.Rows[RowCount].Cells[8].Value = ( Value8 );
//前期比%( Value9 )
if Value8 <> 0 then Value9 = ( ( Value2 - FundFieldValue ) / FundFieldValue * 100 );
DGV0.Rows[RowCount].Cells[9].Value = ( Value9 );
//色変更
ColorSet();
end;{ FQPDataSet2 method }
{ ----- } //色変更
method void ColorSet()
begin
//マイナスは青表示、フォーマットのやり方が分からない為ゼロは背景色で隠すw
if Value1 < 0 then DGV0.Rows[RowCount].Cells[1].ForeColor = color.DodgerBlue else
if Value1 = 0 then DGV0.Rows[RowCount].Cells[1].ForeColor = color.Black;
if Value2 < 0 then DGV0.Rows[RowCount].Cells[2].ForeColor = color.DodgerBlue else
if Value2 = 0 then DGV0.Rows[RowCount].Cells[2].ForeColor = color.Black;
if Value3 < 0 then DGV0.Rows[RowCount].Cells[3].ForeColor = color.DodgerBlue else
if Value3 = 0 then DGV0.Rows[RowCount].Cells[3].ForeColor = color.Black;
if Value4 < 0 then DGV0.Rows[RowCount].Cells[4].ForeColor = color.DodgerBlue else
if Value4 = 0 then DGV0.Rows[RowCount].Cells[4].ForeColor = color.Black;
if Value5 < 0 then DGV0.Rows[RowCount].Cells[5].ForeColor = color.DodgerBlue else
if Value5 = 0 then DGV0.Rows[RowCount].Cells[5].ForeColor = color.Black;
if Value6 < 0 then DGV0.Rows[RowCount].Cells[6].ForeColor = color.DodgerBlue else
if Value6 = 0 then DGV0.Rows[RowCount].Cells[6].ForeColor = color.Black;
if Value7 < 0 then DGV0.Rows[RowCount].Cells[7].ForeColor = color.DodgerBlue else
if Value7 = 0 then DGV0.Rows[RowCount].Cells[7].ForeColor = color.Black;
if Value8 < 0 then DGV0.Rows[RowCount].Cells[8].ForeColor = color.DodgerBlue else
if Value8 = 0 then DGV0.Rows[RowCount].Cells[8].ForeColor = color.Black;
if Value9 < 0 then DGV0.Rows[RowCount].Cells[9].ForeColor = color.DodgerBlue else
if Value9 = 0 then DGV0.Rows[RowCount].Cells[9].ForeColor = color.Black;
end;{ ColorSet method }
{ ----- }
```