THE MONEY MANAGER TRADING STRATEGY

An effective trading strategy is only part of a successfully trading plan. If you want your trading to perpetuate, you better have some form of money management built into your overall trading approach. Money management involves examining the concepts of risk and return in reference to investor preference. The objective is to choose a desired rate of return and then minimize the risk associated with that rate of return. Money management concepts should be used to make the most efficient use of trading capital. We can't emphasize enough the importance of using money management in a trading plan. The Money Manager strategy is a simple system that incorporates and demonstrates some simple money management concepts. The concepts we are presenting go beyond simple profit objectives or protective stops. These ideas fall within the realm of the underlying trading strategy. We go beyond this and move into the areas of capital allocation. The concepts that are presented in this strategy are based on capital preservation and market normalization. We all know what capital preservation is, but some may not understand the concept of market normalization. The ability to diversify equal amounts of capital across a portfolio of different markets is the backbone of any money management scheme. If we want to risk 5% of our equity on soybeans and 5% on Treasury bonds, we need the ability to treat the two markets on apples to apples basis. Most of the time one contract of Treasury bonds exhibits more risk than one contract of soybeans. Since we want to maintain a constant amount of capital to risk on the two markets, we will need to trade less Treasury bonds and more soybeans. Let's say the implied market risk for Treasury bonds is $1000 and $500 for soybeans. If we were risking $2000 on each market, we would then trade 2 contracts of bonds and 4 contracts of soybeans. We are maintaining the same amount of risk by varying the number of contracts for the two markets.

Measuring market risk is the first step in the market normalization process. Money managers use several different measures to monitor market risk: average true range, mean change in closing prices, standard deviation in closing prices, and numerous others. The Money Manager strategy uses the standard deviation in closing prices to calculate market risk.

```
Inputs: initCapital(100000),rskAmount(.02);
Vars: marketRisk(0),numContracts(0);

marketRisk = StdDev(Close,30) * BigPointValue;
```

The StdDev function returns the standard deviation in terms of points, so we multiply by BigPointValue (dollar value of a big point move) to get market risk in terms of dollars. Once we know the market risk, we then can calculate the number of contracts.

```
numContracts = initCapital * rskAmount / marketRisk;
```

For demonstration purposes, let's assume we are trading the Japanese Yen and the market risk is equal to $750. The number of contracts would be calculated by using the formula from above:

```
numContracts = 100000 * .02 /750
numContracts = 2000/750
numContracts = 2.66667
```

Since we can only trade with whole contracts, we round down to the nearest whole number. Since market risk is the denominator in our formula, whenever market risk increases the number of contracts decrease, hence, the risk aversion component of our money management scheme. In similar fashion to the Ghost Trader, the Money Manager was designed as more of a template than an actual trading strategy. We wanted to provide the tools necessary to build a money management platform. The source code for Money Manager follows.

The Money Manager Code

```
{The Money Manager}
{Demonstrates the programming and use of a money management scheme.}
{The user inputs initial capital and the amount he wants to risk on each
trade.}
Inputs: initCapital(100000),rskAmt(.02);
Vars: marketRisk(0),numContracts(0);
marketRisk = StdDev(Close,30) * BigPointValue;
numContracts = (initialCapital * rskAmt) / marketRisk;
value1 = Round(numContracts,0);
if(value1 > numContracts) then
       numContracts = value1 - 1
else
       numContracts = value1;
numContracts = MaxList(numContracts,1); {make sure at least 1 contract is
                                         traded}

Buy("MMBuy") numContracts shares tomorrow at Highest(High,40) stop;
SellShort("MMSell") numContracts shares tomorrow at Lowest(Low,40) stop;

if(MarketPosition = 1) then Sell("LongLiq") next bar at Lowest(Low,20) stop;
if(MarketPosition =-1) then BuyToCover("ShortLiq") next bar at
   Highest(High,20) stop;
```

Overall the logic should be easy to follow. However, there is some code that may not be totally intuitive. The formula that we used to determine the number of contracts doesn't always produce a whole number. Since we are trading

stocks or futures, we can't have a fractional number of contracts (shares). The Round function was used to eliminate the fractional part. The Round function requires two parameters: the value to be rounded and the level of precision. The level of precision parameter relates to how many decimal places are to be rounded to. If we had used one instead of zero, then the number of contracts would have been rounded to the nearest tenth place. We used zero to round to the nearest whole number. If we had passed two as the parameter, then the function would round to the nearest 100th place. Since we developed a risk averse money management model, the number of contracts should never be rounded up (risk exposure is directly proportional to the number of contracts that we are trading). EasyLanguage does not provide a truncation function (a function that eliminates the fractional part of a number), so we used the following code to always round down:

```
value1 = Round(numContracts,0);
if(value1 > numContracts) then
        numContracts = value1 - 1
else
        numContracts = value1;
```

Value1 is assigned numContracts rounded to the nearest integer. If the round function rounds up, then value1 will be greater than the numContracts variable. If value1 is greater than numContracts, we simply subtract one from value1 and reassign the difference to numContracts. If value1 is not greater than numContracts, then the round function rounded down and we can simply reassign numContracts this value.

We introduced the keyword shares in the order placement logic.

```
Buy("MMBuy") numContracts shares tomorrow at Highest(High,40) stop;
SellShort("MMSell") numContracts shares tomorrow at Lowest(Low,40) stop;
```

The keyword *shares* must be used when trading a variable number of contracts or shares. The number of shares (in the form of a variable name or a literal) must follow the Buy/SellShort keyword and precede the keyword shares.

```
Buy("myBuy") 50 shares tomorrow at Open;
```

Or

```
Buy("myBuy") myNumShares shares tomorrow at Open;
```

You can use Money Manager as a platform to further research into other money management schemes. You can get some great money management ideas out of Nauzer J. Balsara's, "Money Management Strategies for Futures Traders", published by John Wiley and Sons in 1992.