

1. 예스스팟이란? .....	5
2. 예스스팟 모니터 .....	6
1) 화면개요.....	6
2) 화면구성 .....	6
3) 스팟전략 적용 .....	7
4) 스팟전략 보안설정 .....	8
5) 스팟전략 배포버전 .....	9
3. 예스스팟 전략 편집기.....	10
1) 기본도구.....	11
2) 전략리스트창.....	12
3) 스크립트 객체창.....	12
4) 사용자정의 모듈.....	13
5) 스크립트 편집창.....	14
6) 객체 속성 .....	15
7) 객체정보창 .....	15
8) 사용자정의 모듈 관리자 .....	16
9) 화면데이터 .....	17
10) 디버깅창 .....	18
11) 오류보고창.....	19
12) 찾기결과창.....	19
12) 기본 인터페이스 .....	20
13) 기본용어 .....	22
4. 스크립트 객체화면 설정 .....	25
1) main객체.....	25
2) 외부변수 .....	26
3) 종목객체 .....	27

4) 계좌객체.....	29
5) 차트객체.....	29
6) 확장차트 객체 .....	31
7) 데이터베이스 객체.....	36
8) 엑셀데이터 객체.....	37
9) DDE 객체.....	38
10) 옵션 객체.....	41
11) 참조데이터 객체 .....	41
12) 그리드 객체 .....	42
13) 객체삭제 .....	44
<b>5. 객체, 메소드(Method), 프로퍼티(Property), 이벤트(Event) 설명 .....</b>	<b>45</b>
1) Main객체.....	45
2) 종목객체.....	60
3) 계좌객체.....	68
4) 차트객체.....	79
5) 확장차트 객체 .....	83
6) 데이터베이스 객체.....	86
7) 엑셀데이터 객체.....	88
8) DDE 객체.....	90
9) Option 객체.....	91
10) 참조데이터객체 .....	108
11) 그리드객체.....	108
11) 체결통보객체 .....	110
12) 주문응답객체 .....	112
13) 완성신호객체 .....	114
14) 미완성신호객체 .....	116
15) 미체결내역객체 .....	116

16) 잔고액체(Balance) .....	118
17) 확장차트 종목설정 객체 (ReqChartItem) .....	119
18) 확장차트 시스템설정 객체 (SystemInfo) .....	122
19) 확장차트 수식적용 Input변수 객체 (YLIInputVar).....	124
20) 확장차트 시스템적용 피라미딩, 비용/수량 설정 객체 (SystemTradeInfo) .....	125
21) 확장차트 시스템적용 강제청산 설정 객체 (SystemStopInfo) .....	128
22) 확장차트 시스템적용 손절매 설정 객체 (StopLoss).....	130
23) 확장차트 시스템적용 목표수익 설정 객체 (StopProfitTarget).....	131
24) 확장차트 시스템적용 최대수익대비하락 설정 객체 (StopTrailing) .....	131
25) 확장차트 시스템적용 최소가격변화 설정 객체 (StopInactivity).....	132
26) 확장차트 시스템적용 당일청산 설정 객체 (StopEndOfDay) .....	133
27) 확장차트 지표 설정 객체 (IndicatorInfo).....	133
28) YesSpot Constants(상수) .....	134
29) 내장 함수 및 객체 .....	140
<b>5. 예스스팟 사용시 주의사항 .....</b>	<b>143</b>
<b>6. 작성예제 .....</b>	<b>145</b>
예제1. 시스템 연동 타종목주문 .....	145
예제2. 복수 종목데이터 이용 .....	156
예제3. 시간정정주문 .....	170
예제4. 취소주문.....	176
예제5. 차트연동주문 .....	186
예제6. 합성선물.....	191
예제7. 주기가 다른 여러 차트를 참조해 주문발생하기.....	200
예제8. 종목객체를 동적으로 생성해서 이용하기.....	207
예제9. 사용자정의 함수 만들기 .....	211
예제10. 사용자정의 함수객체 만들기 .....	213
예제11. 동시호가청산 .....	216

예제12. 특정 가격 이하의 콜/풋 옵션 중 가장 높은 가격의 종목을 선정 .....	218
예제13. 일정주기로 종목검색을 하고 검색된 종목을 매수 .....	222
예제14. 종목검색에서 결과값 이용 .....	227

## 1. 예스스팟이란?

예스스팟은 전문 IT인력을 보유하지 못한 개인투자자나 중소규모의 기관들에게 좀 더 쉽게 접근해 사용할 수 있는 종합 트레이딩 플랫폼을 제공하기 위해 개발 되었습니다.

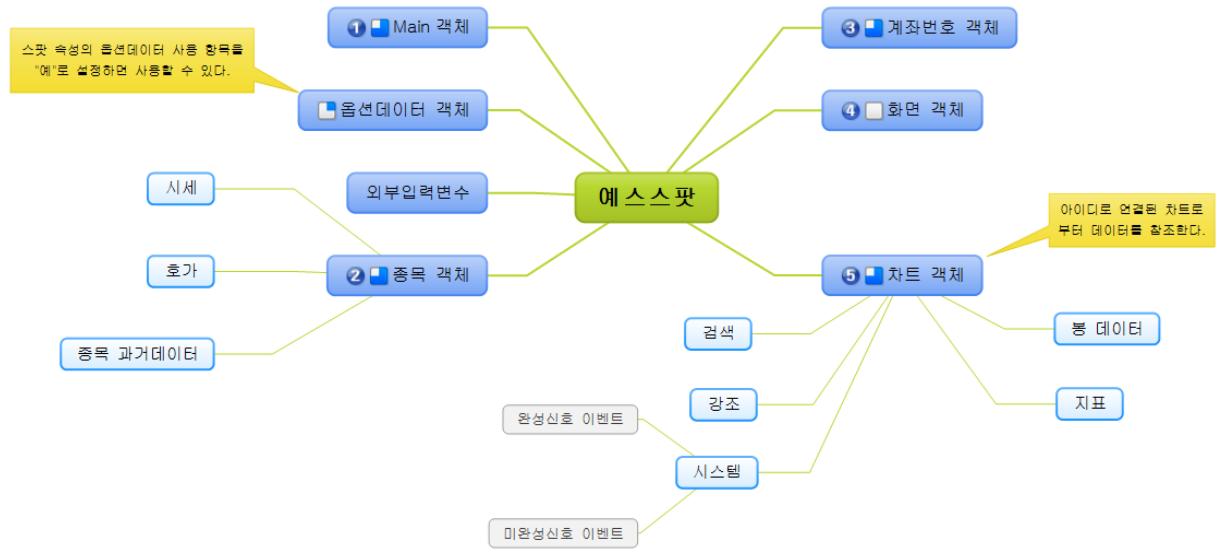
기존에 서비스되는 시스템트레이딩 프로그램의 기능만으로 구현이 어렵거나 불가능한 부분을 확장 보완할 수 있으며 여러 상품을 동시에 거래하거나 다양한 데이터를 참조하는 거래전략에도 이용할 수 있습니다.

예스스팟은 자바스크립트 언어기반에 가격 데이터와 계좌 등 거래에 필요한 제반 데이터를 기본적으로 제공하여 사용자가 데이터의 구축 등에 소비되는 시간 없이 거래전략을 만들어 사용할 수 있도록 지원합니다.

예스스팟은 다음과 같은 전략의 구현이 가능합니다.

(아래 내용은 단순한 예시이며, 이외에도 다양한 전략 구현이 가능합니다.)

- 주식포트폴리오 매매나 페어 트레이딩과 같이 복수 종목의 데이터를 처리하여 전략을 만들고 주문을 실행할 수 있습니다.
- 신호 발생시 주문을 일정한 방법으로 분산하여 실행함으로써 대량 주문의 슬리피지를 줄일 수 있습니다.
- 미완성 신호가 발생될 때 신호를 참조하여 주문을 실행할 수 있습니다.
- 여러 개의 차트에서 나온 신호를 활용하여 자동주문을 실행할 수 있습니다.  
(예:총 5개의 차트 중 3개의 차트에서 매수신호가 발생될 때 매수)
- 타종목, 복수종목, 복수계좌 주문이 가능합니다.  
(예:선물 매수신호 발생시 10만원에 가장 근접한 콜옵션 매수, 풋옵션매도)
- 계좌잔고와 연동하여 자동으로 정정주문이나 취소주문을 실행할 수 있습니다.
- 계좌평가손익이 일정금액 이상이거나 이하일 때 일괄로 청산 주문을 실행할 수 있습니다.
- 보유하고 있는 개별 옵션 종목의 손익이나 그릭 또는 합산된 손익이나 그릭을 이용한 전략 작성과 자동주문이 가능합니다.(옵션 합성매매 기능)



## 2. 예스스팟 모니터

### 1) 화면개요

[예스스팟]은 작성된 전략을 실행할 수 있는 화면입니다.

[예스스팟]에서 작성한 전략들이 리스트되고 리스트에서 전략을 선택해 실행하고 전략에서 발생된 주문들의 내역들을 모니터 할 수 있습니다.

기본메뉴 [시스템트레이딩 → 예스스팟] 또는 화면번호 입력란에 6131을 입력하시면 예스스팟이 실행됩니다.

### 2) 화면구성



예스스팟은 전략리스트, 실행전략리스트, 주문모니터 화면으로 구성되어 있습니다.

① 전략리스트

예스스팟 편집기에서 작성한 전략을 리스트 해 보여주는 화면입니다.

② 실행전략리스트

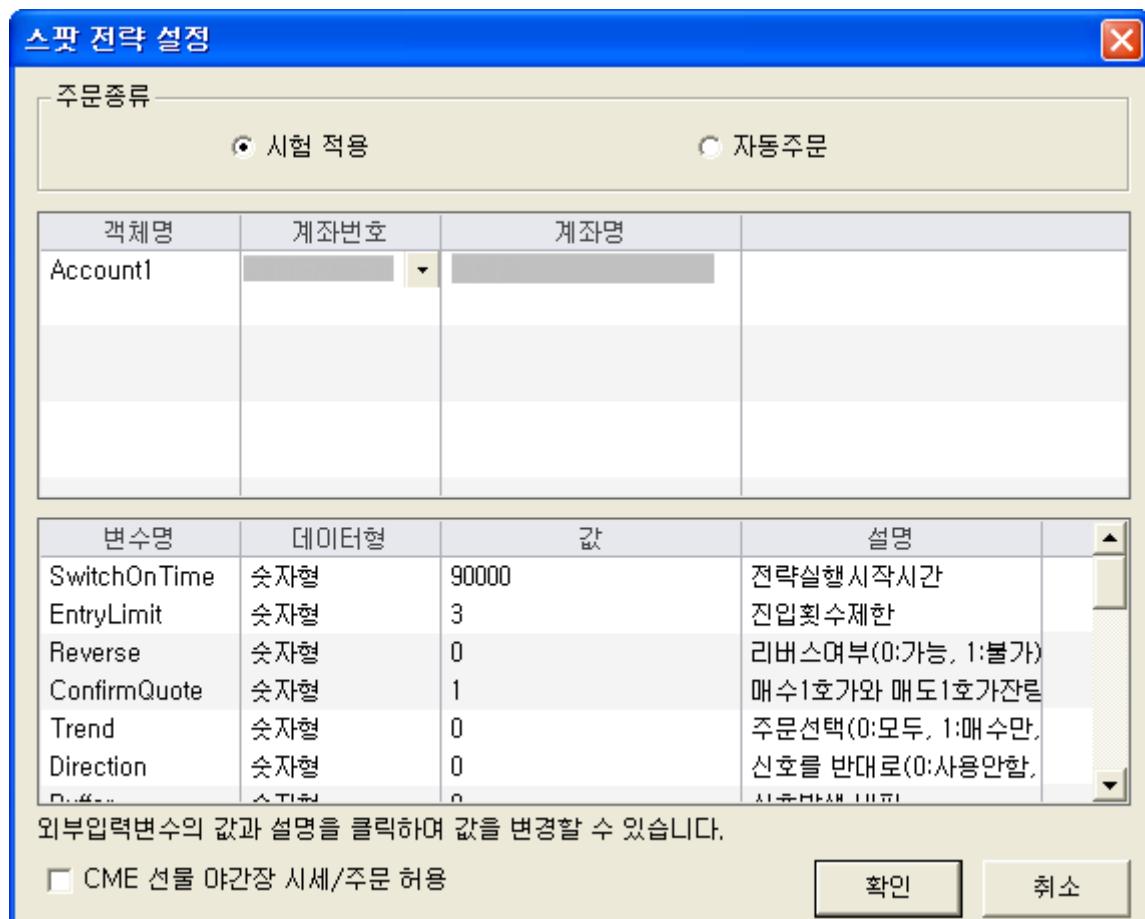
스팟전략 중 실행 중인 전략을 리스트 해 보여주는 화면입니다.

③ 주문모니터

실행된 전략에서 발생한 주문에 대한 체결/미체결 내역을 보여주는 화면입니다.

### 3) 스팟전략 적용

좌측 전략리스트에서 실행할 전략을 더블클릭하면 [스팟 전략 설정]화면이 나타납니다.



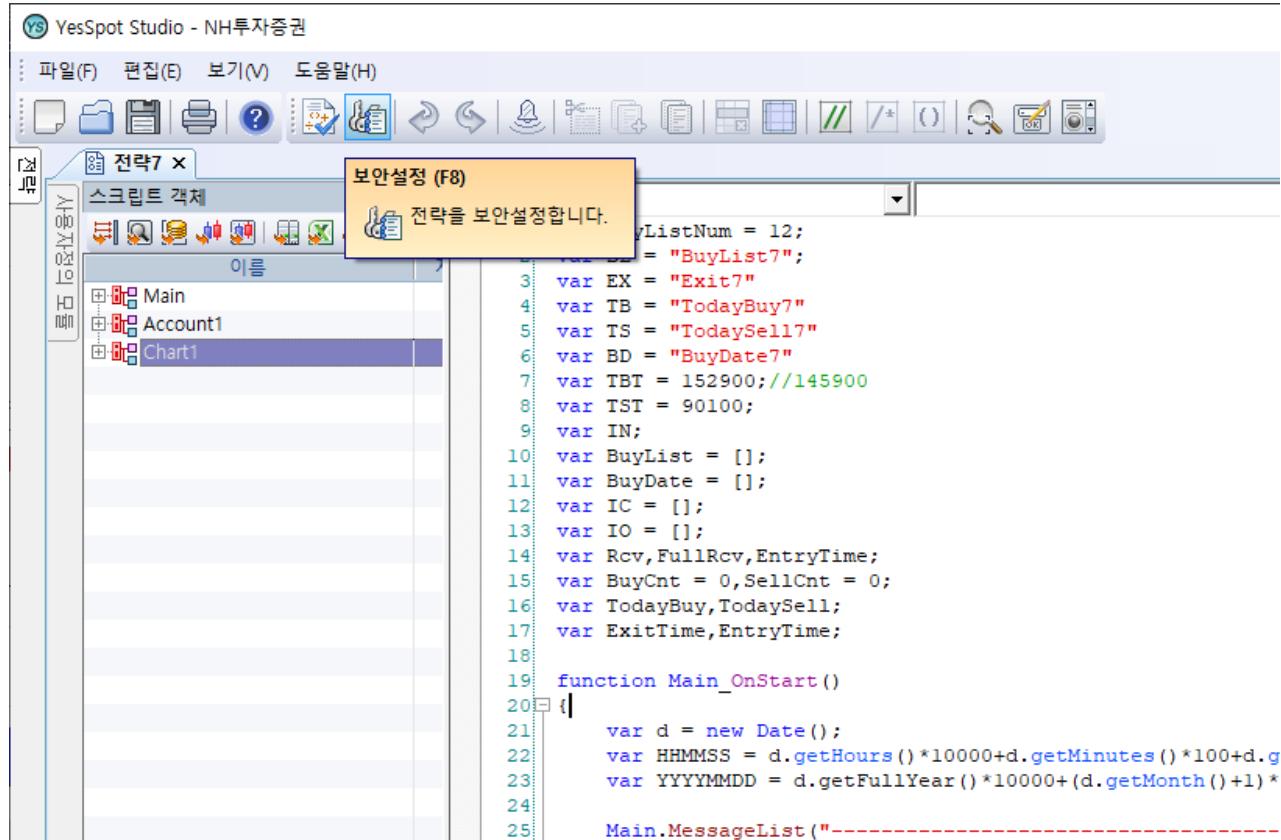
※ 야간선물의 시세를 받거나 주문을 내기 위해서는 [CME 선물 야간장 시세/주문 허용]을 체크해야 합니다.

주문종류 및 외부변수 설정을 완료하고 확인 버튼을 누르시면 전략이 실행전략리스트에 추가되며 실행되게 됩니다.

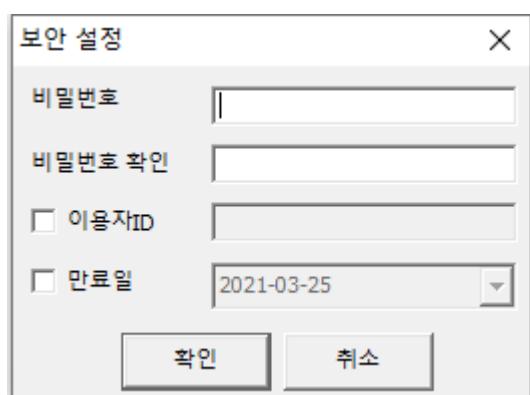
실행 중 주문이 발생하면 주문모니터에서 체결/미체결 결과를 확인하실 수 있습니다.

#### 4) 스팟전략 보안설정

예스스팟스튜디오에서 스팟수식을 열 때 비밀번호 확인을 받는 기능입니다.



스팟수식을 작성완료하고 저장 후에 상단의 보안설정이나 F8키를 누르면 보안설정 화면이 나타납니다.



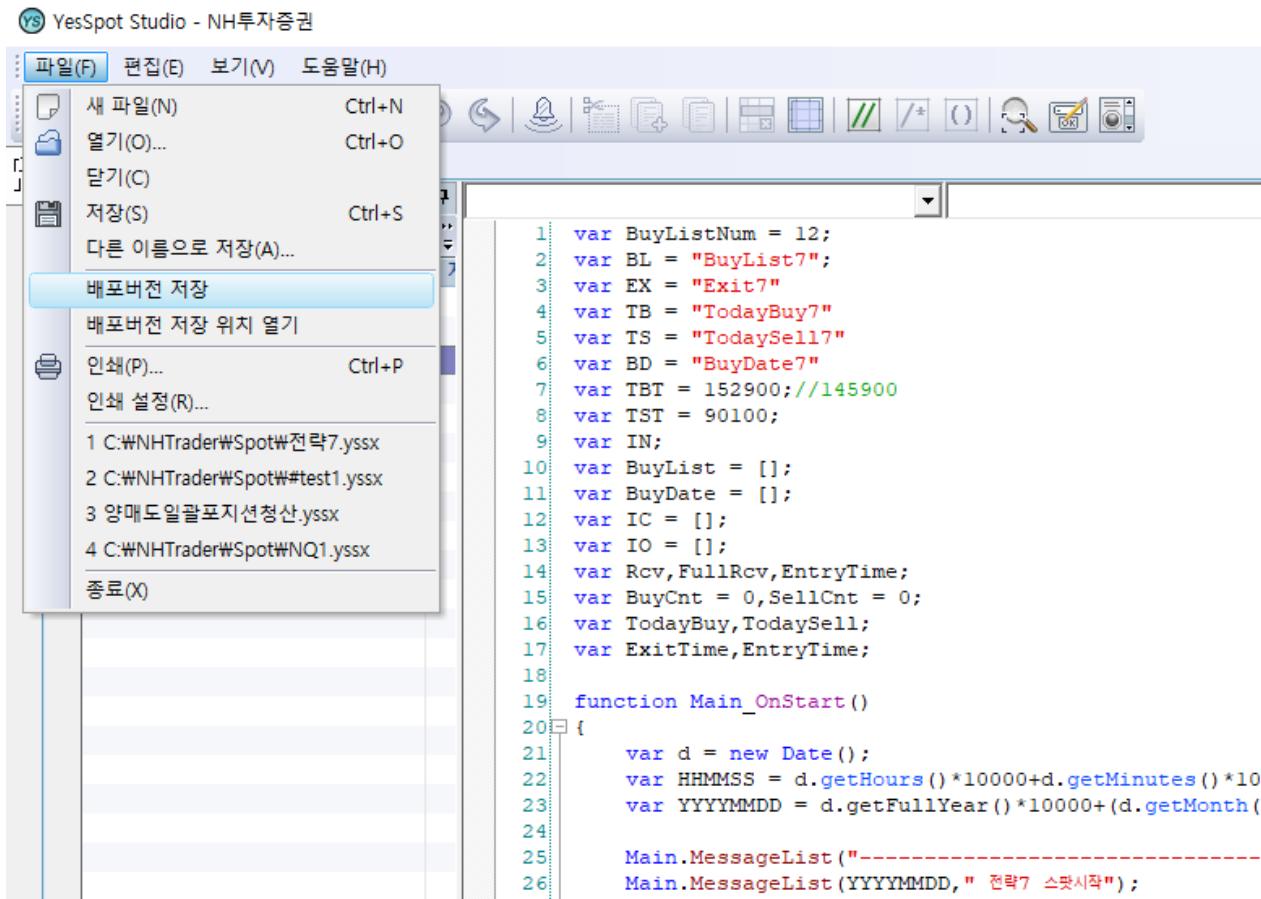
비밀번호 : 비밀번호를 지정합니다. 예스스팟스튜디오에서 스팟수식을 열 때 비밀번호 확인창이 나타납니다.

이용자ID : 지정 ID로 로그인된 경우에만 스팟실행을 할 수 있습니다.

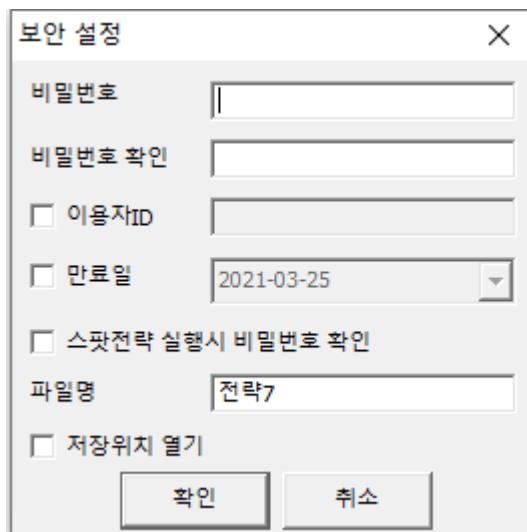
만료일 : 지정일까지만 스팟실행이 가능합니다.

## 5) 스팟전략 배포버전

배포버전은 수식을 암호화하여 편집기에서 볼수 없고 [6131]예스스팟화면에서 실행만 가능하게 만든 파일입니다.



스팟수식을 작성완료하고 저장 후에 상단의 파일메뉴에서 [배포버전 저장]을 누르면 보안설정 화면이 나타납니다.



비밀번호 : 비밀번호를 지정합니다.

이용자ID : 지정 ID로 로그인된 경우에만 스팟실행을 할 수 있습니다.

만료일 : 지정일까지만 스팟실행이 가능합니다.

스팟전략 실행시 비밀번호 확인 : 체크하면 스팟적용시 비밀번호를 입력해야 됩니다.

파일명 : 파일명을 지정합니다.

저장위치열기 : 체크하면 배포버전이 저장된 폴더를 자동으로 열어 배포버전파일을 볼 수 있습니다.

\* 배포버전 파일은 스팟편집이 불가능하게 만들어지는 파일입니다.

\* [6131]예스스팟화면에서 적용만 가능하고 스팟스튜디오에서 파일명이 나타나지 않습니다.

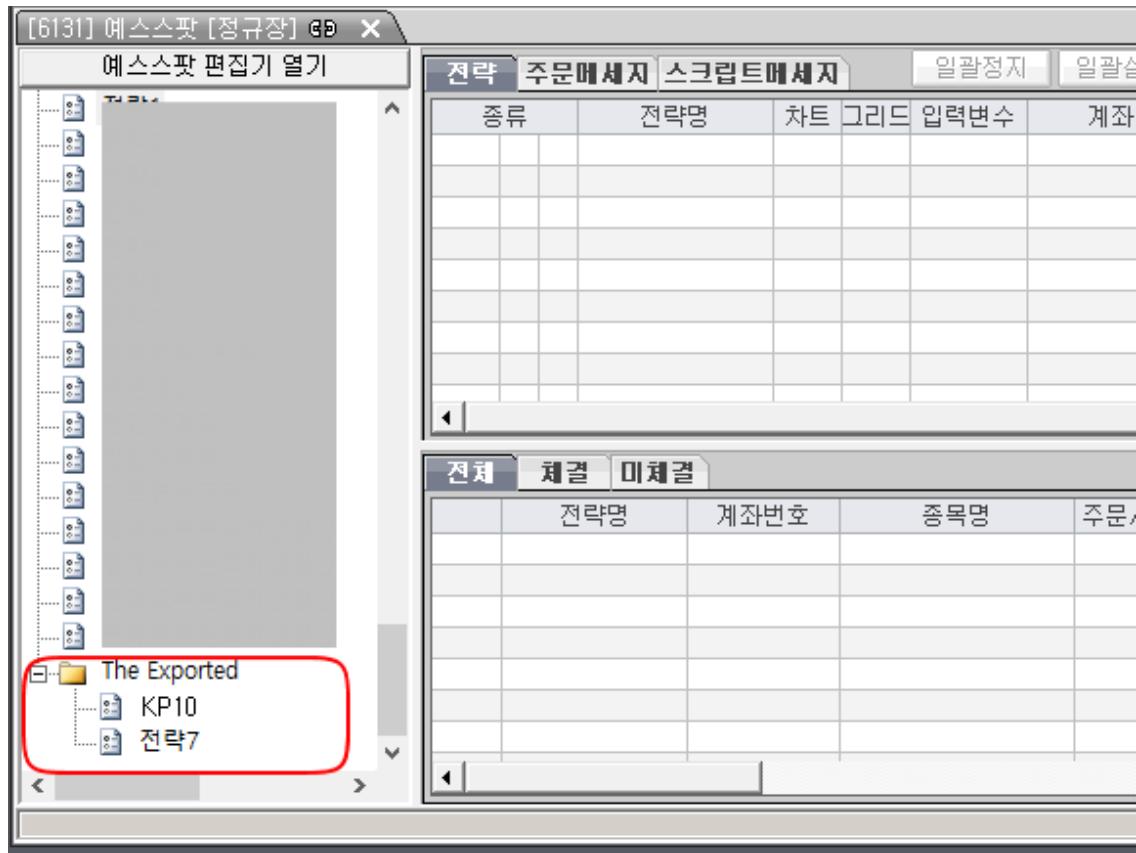
\* 배포버전 파일은 프로그램 설치폴더 안의 Spot폴더 Export폴더에 생성됩니다.

C:\W프로그램설치폴더\WSpot\WExport

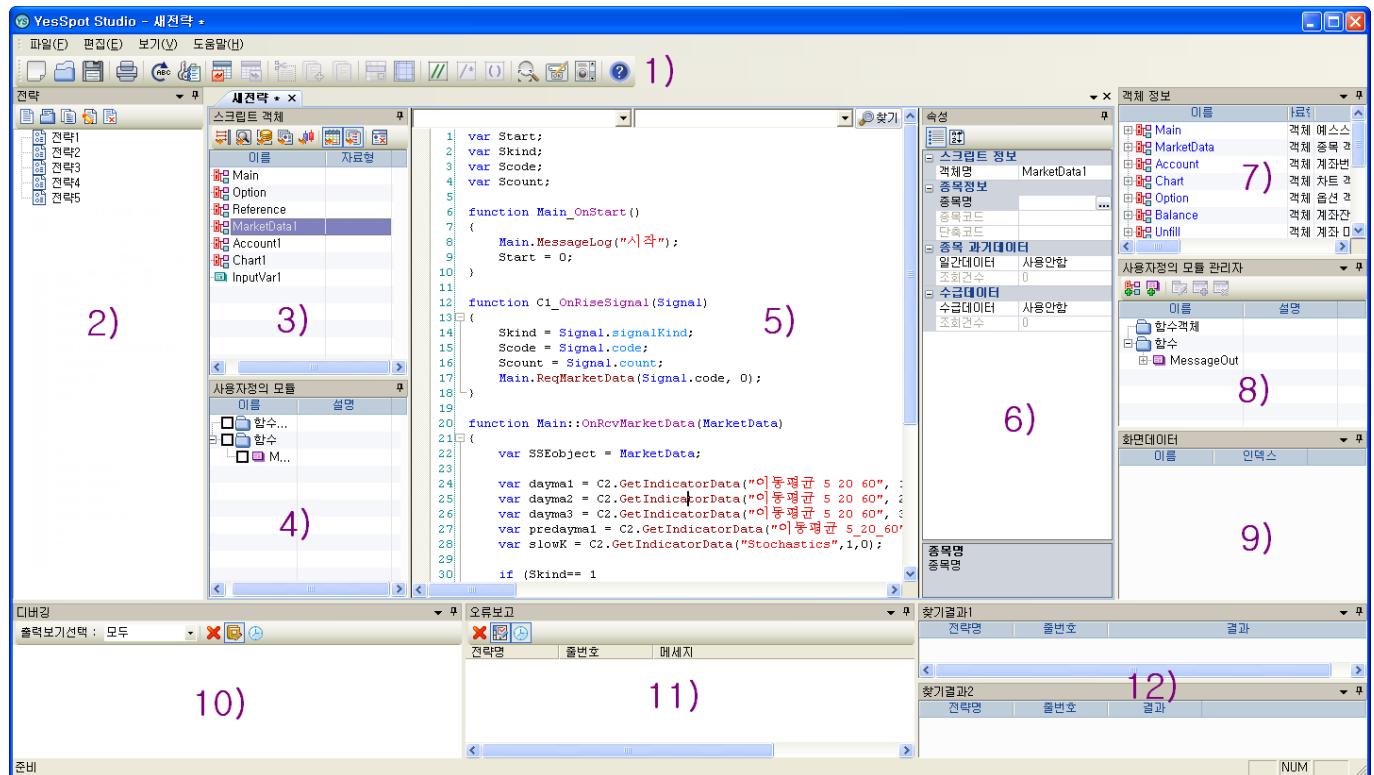
\* 스팟수식의 기본확장자는 파일명.yssx이고 배포버전은 파일명.ysse입니다.

\* 배포버전을 받는 분은 Spot폴더에 파일을 옮긴 후에 사용하시면 됩니다.

\* 배포버전전략은 [6131]예스스팟 화면의 전략목록 하단의 exported폴더에서 찾을 수 있습니다.



### 3. 예스스팟 전략 편집기



예스스팟 전략을 작성하는 화면입니다.

기본메뉴 [시스템트레이딩 → 예스스팟 편집기] 선택하거나 또는 [6132] 예스스팟 화면에서 [예스스팟 편집기 열기]

를 선택하거나 화면번호 입력란에 6132를 입력하면 예스스팟 편집기가 생성됩니다.

예스스팟 편집기는 크게 11가지 화면으로 구성되어 있습니다.

### 1) 기본도구



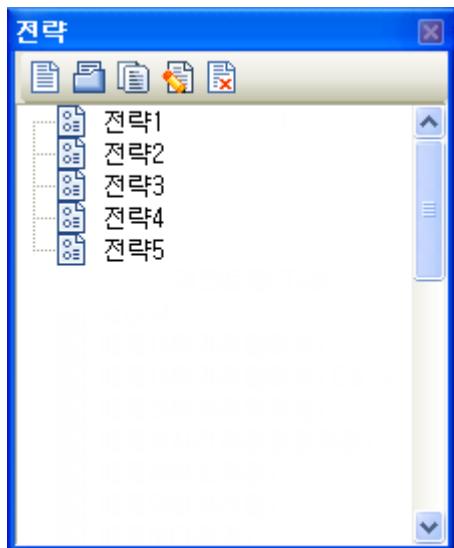
- : 새로운 전략 편집기를 생성합니다.
- : 전략리스트에서 선택한 전략을 엽니다.
- : 현재 편집 중인 전략을 저장합니다.
- : 전략식 내용을 프린트 합니다.
- : 전략식의 기본적인 문법검증을 합니다. 문법검증 내용은 오류보고화면에 출력됩니다.
- : 보안설정
- : 이전 상태로 되돌리기
- : 되돌리기 전의 상태로 복구하기
- : 전략식 내에 지정한 영역을 잘라내기
- : 전략식 내에 지정한 영역을 복사하기
- : 잘라내기 혹은 복사한 영역을 붙여넣기
- : 전략식 내에 선택한 영역을 삭제하기

-  : 전체 내용을 선택
-  : 한 줄 주석처리
-  : 여러 줄 주석처리
-  : 선택영역 팔호치기(소괄호)
-  : 전략 식에서 지정한 문자열 찾기(Ctrl + F)
-  : 전략 식에서 지정한 문자열을 다른 문자열로 바꾸기(Ctrl + H)
-  : 전체 전략식 내에서 지정한 문자열 찾기( 찾기결과 화면에 표시)
-  : 편집기 정보

## 2) 전략리스트창

전략리스트화면은 예스스팟 편집기에서 작성한 스팟전략들의 목록을 보여주는 화면입니다.

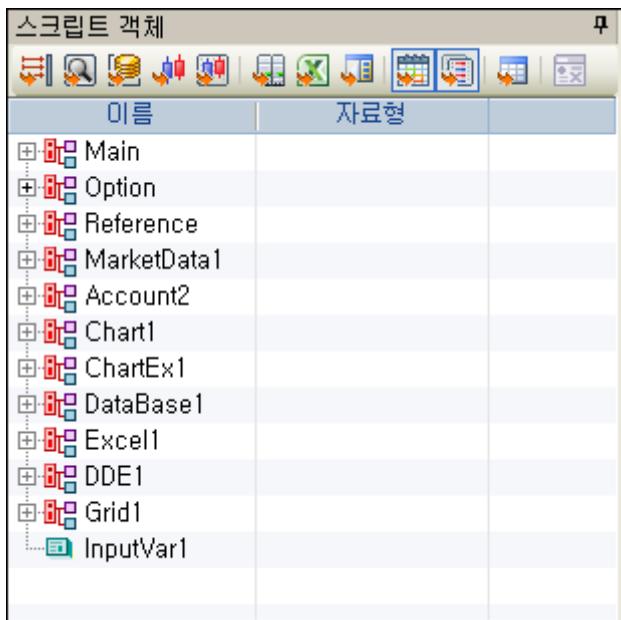
목록에서 전략을 선택하면 해당 전략을 열어 편집할 수 있고 전략의 복사 및 삭제 등의 관리가 가능합니다.



-  : 새전략 만들기
-  : 전략 열기
-  : 전략 복사
-  : 전략명 변경
-  : 전략 삭제

## 3) 스크립트 객체창

객체 목록창은 스크립트에서 사용할 객체나 외부변수를 추가하고 설정하는 화면입니다.



스크립트 객체 - 상단 아이콘		
: 외부변수 추가	: 종목 객체 추가	: 계좌 객체 추가
: 차트 객체 추가	: 확장차트 객체 추가	
: 데이터베이스 객체 추가	: 엑셀데이터 객체 추가	: DDE객체 추가
: 옵션 객체 추가	: 참조 객체 추가	: 그리드객체 추가
: 선택 객체 삭제		

스크립트 객체화면에서 각 객체는 로 표시되며 특성에 따라 각각의 메소드()나 프로퍼티()나 이벤트()를 포함하고 있습니다.

이벤트는 현재 스크립트에서 사용 중인 이벤트이면 로 표시되고 사용되지 않는 이벤트이면 로 표시됩니다.

외부변수는 로 표시됩니다.

#### 4) 사용자정의 모듈

사용자가 직접 [사용자정의 모듈 관리자]에서 작성한 함수와 함수객체를 이용할 수 있는 화면입니다.

목록에서 사용하고자 하는 함수와 함수객체를 체크 후 사용하시면 됩니다.

사용자정의 모듈	
이름	설명
함수객체	UserObject3 매개변...
함수	NotifySignal

### 5) 스크립트 편집창

스크립트 편집창은 스팟전략을 작성하는 창입니다.

The screenshot shows the MetaTrader 5 Script Editor interface. At the top, there are three tabs: '사용중인 이벤트 목록' (Used Events List), '문자열 찾기 목록' (String Search List), and '찾기' (Search). Below these tabs, the main area displays a script with several functions:

```

function Main_OnStart()
function C1_OnRiseSignal(Signal)
function Main_OnStart()
    function Main_OnStart()
        Main_MessageLog("시작");
        Start = 0;
    }
    ...
    if (Signal.signalKind == 1
        && dayma1 > dayma2 && dayma2 > dayma3
        && dayma1 > predayma1
        && slowK <= 30)
    ...
    if (Signal.signalKind == 2 && Start == 1)
    {
        A1.OrderSell(Signal.code, Signal.count, SSE.Bid(2), 0);
    }
}

```

Annotations with arrows point from the tabs to their respective sections in the editor. A large arrow points from the '찾기' tab to the search bar in the top right of the editor window. Another arrow points from the '문자열 찾기 목록' tab to the search results list below the search bar.

※ 사용중인 이벤트 목록 : 스크립트 수식 안에 현재 사용중인 이벤트들을 확인 가능하며 선택하면 해당 이벤트 사용지점으로 이동됩니다.

※ 문자열 찾기 목록 : 스크립트 수식 내에 지정한 단어를 찾습니다.

※ 블록 열기/닫기 : 중괄호로 블록으로 지정한 영역을 단기하면 수식을 간결하게 보실 수 있습니다.

## 6) 객체 속성

속성창은 객체목록에 추가한 객체의 속성을 설정하고, method, event, Property등의 도움말 등의 정보를 보실 수 있는 화면입니다.



## 7) 객체정보창

예스스팟에서 제공되는 모든 객체와 객체에 포함된 이벤트, 메소드, 프로퍼티, 상수에 대한 정보를 간단한 도움말과 함께 보여주는 화면입니다.

이름	자료형	설명
Main	객체	예스스팟 스크립트의 기본 객체
Account	객체	계좌번호 객체
Balance	객체	계좌잔고 객체
Chart	객체	연결 차트 객체. 이미 생성되어 있는 차트에 ID를 지정해 인터페이...
ChartEx	객체	차트를 지정한 설정에 맞게 새로 생성하여 차트객체가 참조합니다.
DataBase	객체	ODBC관리자를 통하여 데이터베이스에 연결할 객체입니다.
Excel	객체	예스스팟과 엑셀을 연동하기 위한 객체
IncompleteSignal	객체	미완성신호 객체
IndicatorInfo	객체	차트에 적용할 지표의 정보를 설정하기 위한 객체
Library	객체	서비스 준비중입니다. DLL파일로 만들어진 라이브러리를 객체로 ...
MarketData	객체	종목 객체
NotifyFill	객체	주문체결통보 객체
Option	객체	옵션 객체
OrderResponse	객체	주문접수응답 객체
Reference	객체	참조 객체
ReqChartItem	객체	차트에 적용할 종목을 설정하기 위한 객체
Signal	객체	신호 객체. 차트에서 발생한 신호에 대한 정보를 담고 있는 객체입...
StopEndOfDay	객체	당일 강제 청산 객체
StopInactivity	객체	최소가격변화 강제청산 객체
StopLoss	객체	손절매 강제청산 객체
StopProfitTarget	객체	목표수익 강제청산 객체
StopTrailing	객체	최대수익대비하락 강제청산 객체
SystemInfo	객체	차트에 적용할 시스템의 정보를 설정하기 위한 객체
SystemStopInfo	객체	시스템을 차트에 적용할 때 필요한 강제청산 정보를 설정하기 위한...
SystemTradeInfo	객체	시스템을 차트에 적용할 때 필요한 비용, 수량, 피라미딩 여부 등의...
Unfill	객체	계좌 미체결내역 객체
YLIInputVar	객체	예스랭귀지 수식의 Input 변수를 위한 객체
YesSpot Constants		

객체 정보 화면데이터

#### 8) 사용자정의 모듈 관리자

사용자가 직접 함수나 함수객체를 만드실 수 있는 화면입니다. 사용자 정의 함수나 함수객체를 이용하시면 코드를 간결하게 하실 수 있습니다.

사용자정의 모듈 관리자	
이름	설명
함수객체	
Sign...	차트 신호 정보
함수	
Msg...	차트 완성신호 ...
Notif...	

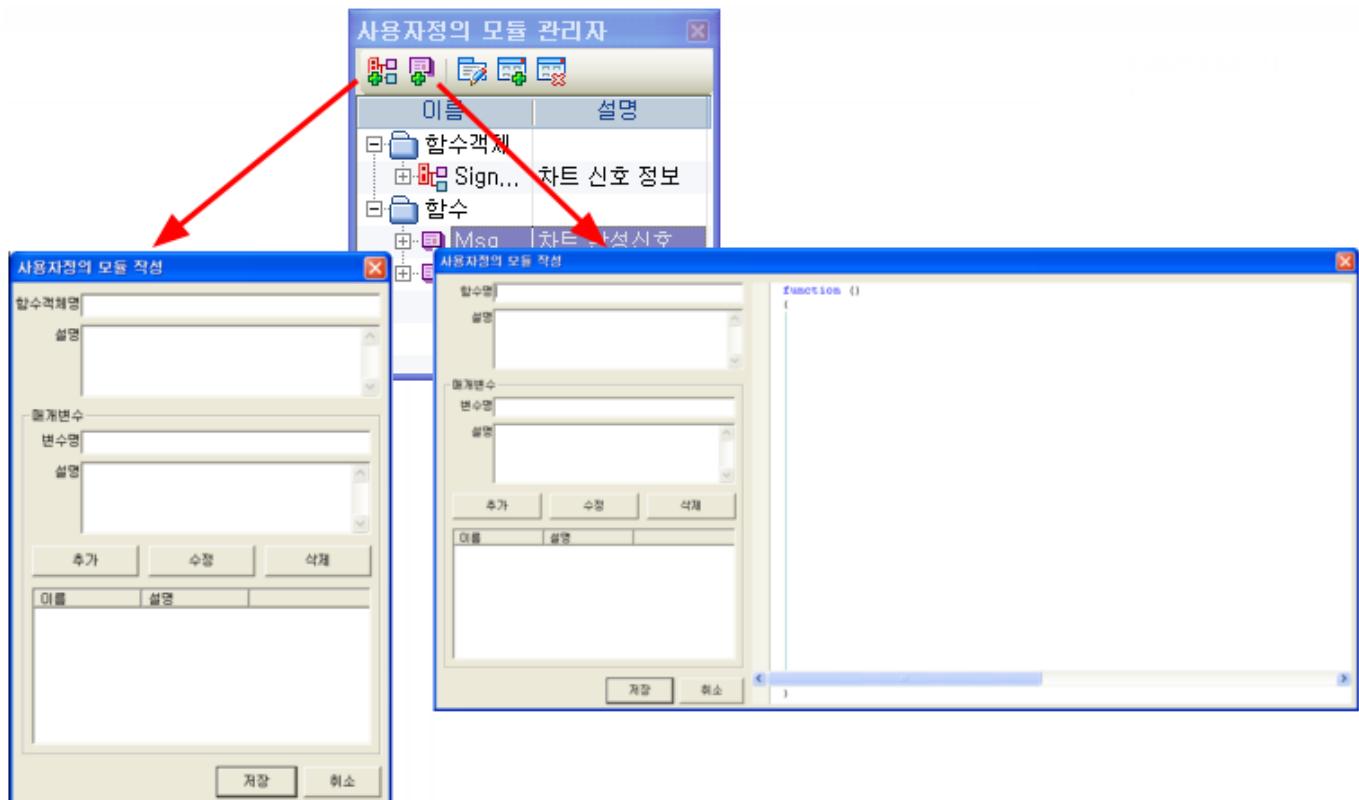
 : 사용자정의의 함수객체 만들기

 : 사용자정의의 함수 만들기

 : 모듈 편집

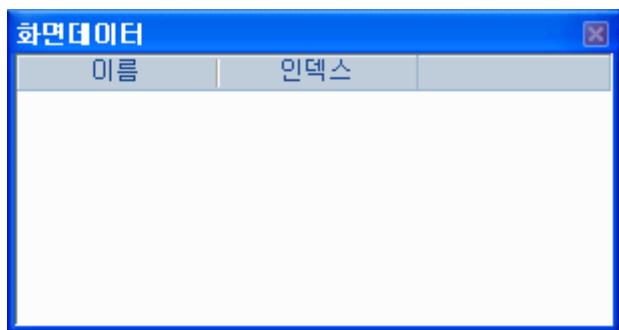
 : 모듈 복사

 : 모듈 삭제



작성된 사용자정의의 함수와 함수객체는 사용자정의의 모듈화면에 나타납니다.

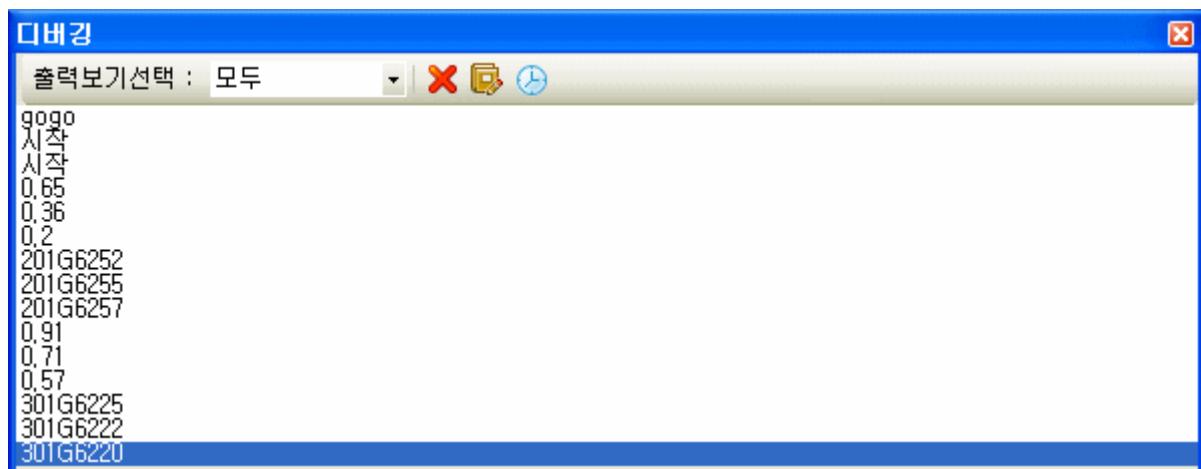
## 9) 화면데이터



화면 객체의 목록이 표시됩니다.

#### 10) 디버깅창

디버깅창은 실행 중인 스팟전략에서 MessageLog로 의해 출력된 내용이 표시됩니다..



출력보기선택에서 전략별 출력 내용을 지정할 수 있습니다.

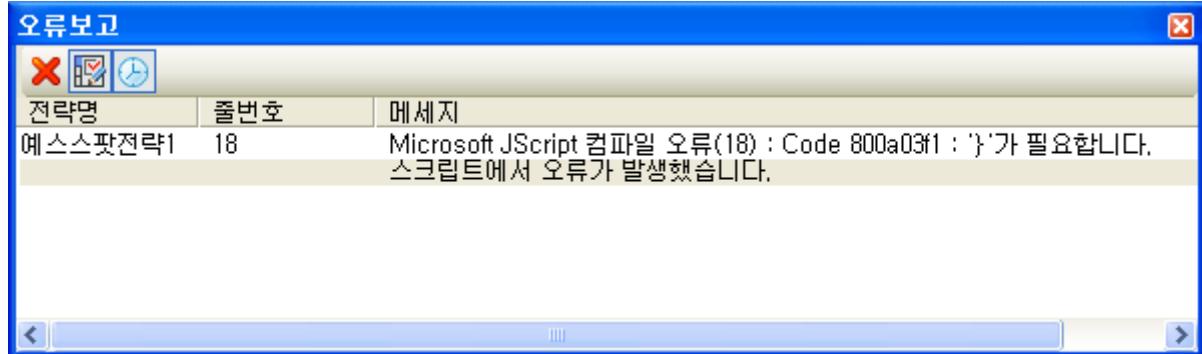
: 전체 메시지 삭제

: 마지막 출력내용 선택

: 메시지 알림

## 11) 오류보고창

오류보고창은 기본문법에 대한 오류나 스크립트 실행 중에 발생된 오류내용을 표시하는 화면입니다.



: 전체 오류내용 삭제

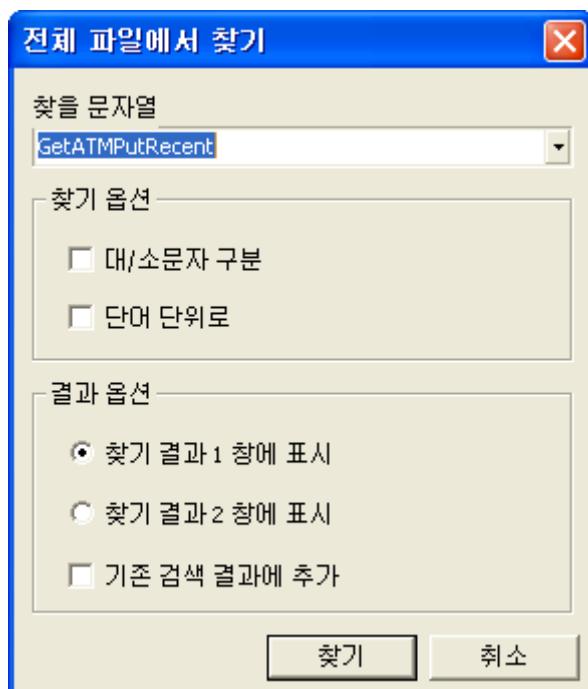
: 마지막 오류내용 선택

: 오류 알림

## 12) 찾기결과창

기본도구에서 파일에서 찾기()를 실행하고 검색 하면 그 결과를 표시해 주는 화면입니다.

찾기결과창1과 찾기결과창2 중 하나를 선택해서 출력할 수 있습니다.



전략명	줄번호	결과	
예제3(타종목주문)	61	ooSellSymbol1 = Option, GetATM...	
	62	ooSellSymbol2 = Option, GetATM...	
	63	ooSellSymbol3 = Option, GetATM...	
예제5(옵션 10이하 ...	22	ooPutCode = Option, GetATMPutR...	

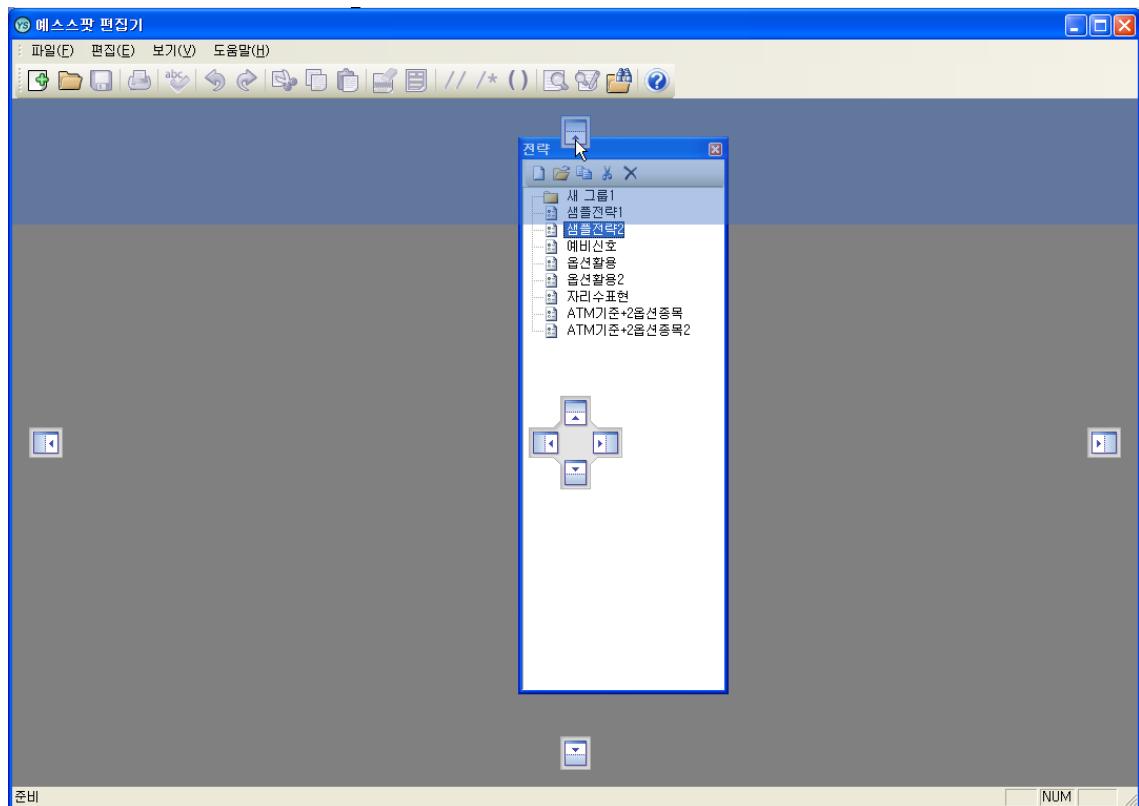
찾기결과1 찾기결과2

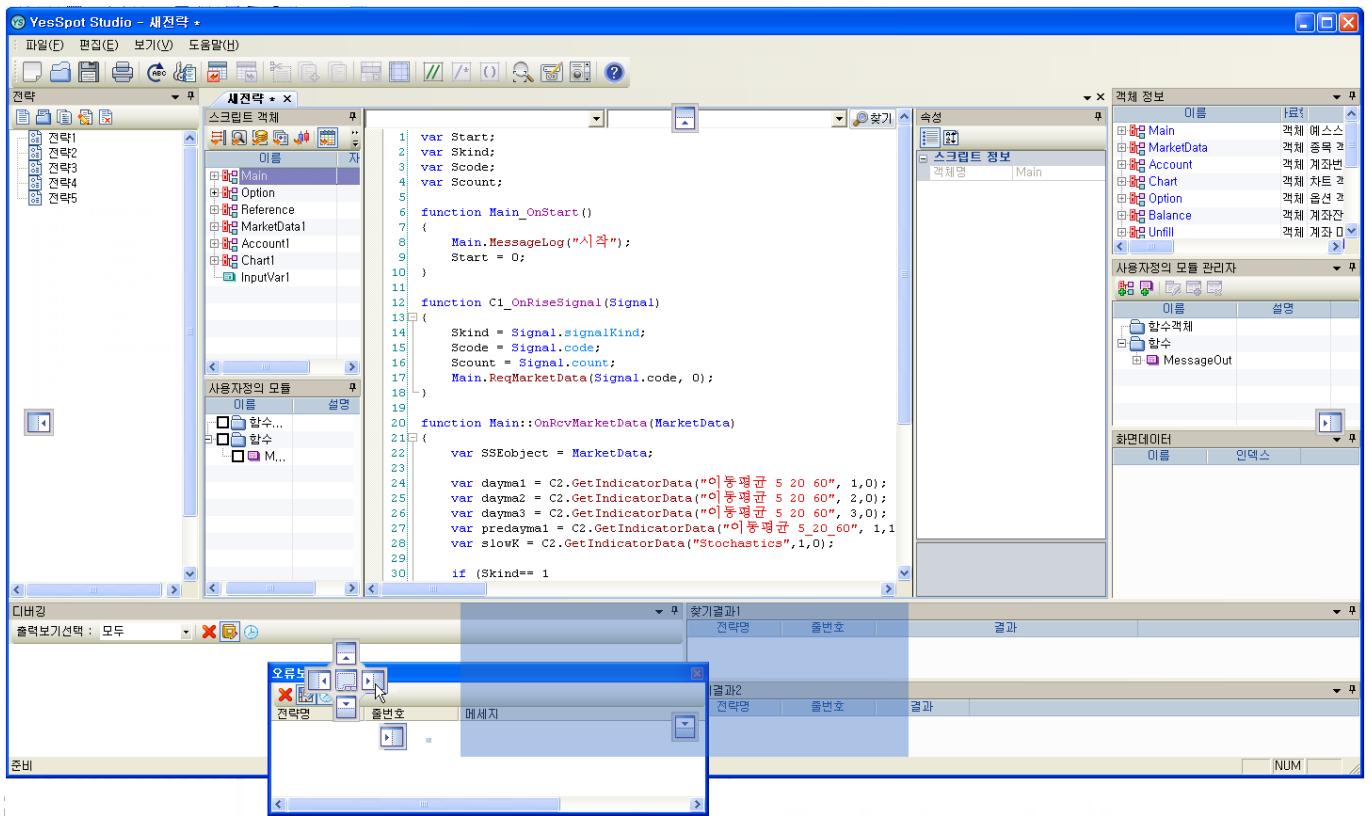
## 12) 기본 인터페이스

### ① 도킹 and 플로팅

도킹은 특정화면을 메인화면 틀 안쪽으로 붙이는 기능이며 플로팅은 화면이 떠 있는 상태를 말합니다.

예스스팟 편집기의 각 화면은 각 화면을 플로팅 상태로 배치해서 사용하실 수 있고 예스스팟 편집기의 틀 안에서 상/하/좌/우에 도킹해서 사용자의 기호에 맞게 화면을 구성해서 사용할 수 있습니다.





1. 플로팅되어 있는 화면을 마우스로 선택하면 도킹 가능한 위치가 표시됩니다.
2. 각 도킹 위치표시에 화면을 드래그하여 이동하시면
3. 도킹 될 영역을 파란색으로 표시해 주게 됩니다.
4. 원하시는 도킹 위치표시에 화면을 드롭하면 됩니다.
5. 도킹 된 화면은 화면 이름을 더블클릭하면 다시 플로팅 상태가 됩니다.

## ② 화면 숨기기

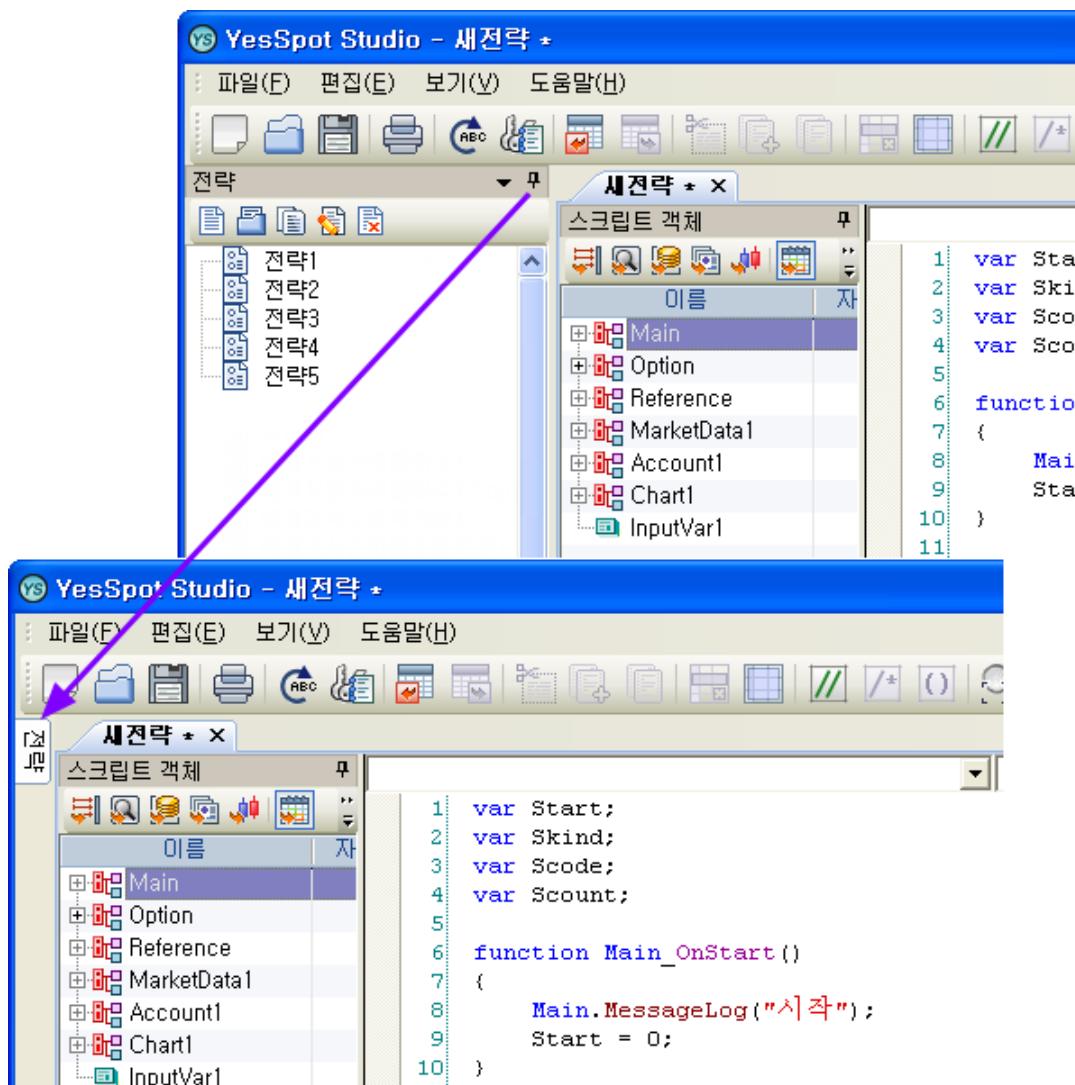
각 화면의 상단의 버튼은 화면을 숨기는 기능입니다.

화면이 플로팅 상태일 때는 사용할 수 없고 도킹 된 화면에서만 사용할 수 있습니다.

을 클릭하면 해당 화면의 이름만 클립형태로 남게 되고 화면이 숨겨지게 됩니다.

클립에 마우스를 가져가시면 해당 화면이 나타나게 됩니다.

자주 사용하지 않는 화면은 자동 숨김 처리하면 좀더 편집기를 넓게 사용할 수 있습니다.



### 13) 기본용어

#### ① 객체

객체란 동일한 목적을 위해 모인 데이터들의 집합입니다.

예를 들면, 주식이나 선물과 같은 하나의 종목에서 현재가, 시가, 고가, 저가 등을 그 종목의 시세입니다.

이런 일련의 데이터들을 모아놓은 것이 하나의 객체이며 현재가, 시가, 고가, 저가 등을 그 객체의 소속된 요소들이 됩니다. 이런 요소들은 그 특징을 알려주는 것은 Property(특성)가 되고 그 객체가 할 수 있는 기능이나 동작이 Method(함수)가 되고, 해당 객체에서 특정 사건이 발생한 것을 알려주는 것이 이벤트(Event)가 됩니다.

예스스팟 스크립트에서는 여러 객체가 제공됩니다.

스크립트 객체화면에서 추가 후 사용하는 객체가 있고 스크립트 내에서 불러와 사용되는 객체가 있습니다.

스크립트 객체화면에 추가해 사용하는 객체

- ※ Main 객체 / 종목 객체 / 계좌 객체 / 차트 객체 / 확장차트 객체
- ※ 데이터베이스 객체 / 엑셀데이터 객체 / 옵션데이터 객체 / 참조데이터 객체

스크립트 내에서 불러와 사용하는 객체

- ※ 체결통보 객체/ 주문응답 객체 / 완성신호 객체 / 미완성신호 객체 / 미체결내역 객체 / 잔고 객체
- ※ 확장차트 종목설정 객체 / 확장차트 지표설정 객체 / 확장차트 시스템설정 객체
- ※ 확장차트 강제청산설정 객체 / 확장차트 손절매설정 객체 / 확장차트 목표수익설정 객체
- ※ 확장차트 최대수익대비하락설정 객체/ 확장차트 당일청산설정 객체 / 확장차트 최소가격변화설정 객체
- ※ 확장차트 기타설정(비용수량, 피라미딩) 객체/ 확장차트 예스랭귀지 Input변수 객체
- ※ DLL Library 객체

스크립트 객체화면에 추가해 사용하는 개체 중에서는 Main과 Option객체, 참조데이터객체와 같이 각각의 고정된 이름으로만 스크립트 객체화면에 등록해서 사용할 수 있는 개체가 있고 종목객체, 계좌객체, 차트 객체, 화면객체등과 같이 사용자가 이름을 지정하여 스크립트 객체화면에 추가한 후 사용할 수 있는 객체도 있습니다.

## ② 이벤트(Event), 메소드(method), 프로퍼티(Property)

객체는 특성에 따라 다음과 같이 각각의 메소드나 프로퍼티, 이벤트를 포함 하고 있습니다.

**스크립트 객체**

이름	자료형
Main	
GetOrderCode	문자열
GetUserValue	문자열
KillTimer	없음
MessageBox	없음
MessageLog	없음
OrderBuy	정수
OrderCancel	정수
OrderReplace	정수
OrderReplacePrice	정수
OrderSell	정수
Pause	없음
Play	없음
RemoveMarketData	없음
ReqMarketData	없음
SetTimer	없음
SetUserValue	문자열
OnNotifyFill	없음
OnOrderResponse	없음
OnRcvMarketData	없음
OnStart	없음
OnTimer	없음
OnUpdateAccount	없음
OnUpdateMarket	없음

**스크립트 객체**

이름	자료형
Option	
RiskFreeRate	실수
Volatility	실수
GetAsk	실수
GetAskByCode	실수
GetAskAmount	정수
GetAskAmountBy...	정수
GetAskCount	정수
GetAskCountByC...	정수
GetAskTotalAmount	정수
GetAskTotalAmo...	정수
GetAskTotalCount	정수
GetAskTotalCoun...	정수
GetATMCallRecent	문자열
GetATMPutRecent	문자열
GetBid	실수
GetBidByCode	실수
GetBidAmount	정수
GetBidAmountBy...	정수
GetBidCount	정수
GetBidCountByC...	정수
GetBidTotalAmount	정수
GetBidTotalAmou...	정수
GetBidTotalCount	정수
GetBidTotalCount...	정수

이벤트는 특정한 동작이 발생했다는 신호를 가리킵니다.

시세를 수신 받거나 체결통보를 받거나 주문응답을 받거나 하는 등 일련의 사건이 이벤트가 됩니다.

에스스팟은 이러한 사건들이 발생할 때마다 구현된 스크립트(이벤트)를 통해 사용자에게 알려주고 사용자는 처리해야 하는 동작을 기술(이벤트에 실행부를 구현)해서 전략을 구현해 가면 됩니다.

이벤트는 `function 객체명_이벤트명([[매개변수], …]) { }` 와 같이 스크립트에 기술하고 중괄호 안에 해당 이벤트가 발생했을 때 구현하고자 하는 내용을 사용자가 기술하면 됩니다.

Chart1객체에서 완성신호 이벤트가 발생하고 완성신호의 종류가 매수진입(Buy)이면

디버깅 창에 “신호완성”이라는 메시지를 출력하는 식을 작성한다면 아래와 같이 작성됩니다.

```
function Chart1_OnRiseSignal(Signal)
{
    if (Signal.signalKind == 1 )
```

```

{
    Main.MessageLog("신호완성/-"+Signal.signalKind);
}

```

프로퍼티와 메소드는 점 연산자를 통해 사용자가 호출하여 사용하는 데이터입니다.

따라서 프로퍼티나 메소드는 예스스팟에서 정의된 기능 및 데이터를 호출하는 개념으로 스크립트에서 사용할 때는 “Account1.number”와 같이 점 연산자의 왼쪽에는 객체가, 오른쪽에는 프로퍼티나 메소드를 지정하여 특정 값을 참조하거나 호출 시 넘겨받은 파라미터에 따라 주문실행 등 사전에 지정된 기능을 수행합니다.

Signal.signalKind → 완성신호객체(Signal)의 신호종류

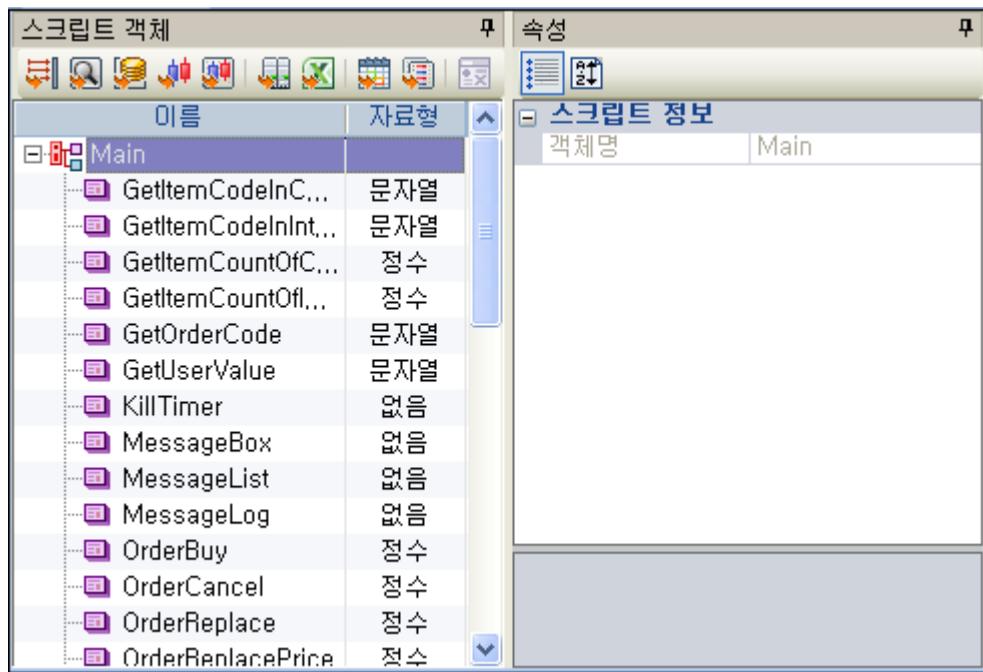
OrderResponse.orderNum → 주문응답객체(OrderResponse)의 주문번호

#### 4. 스크립트 객체화면 설정



##### 1) main객체

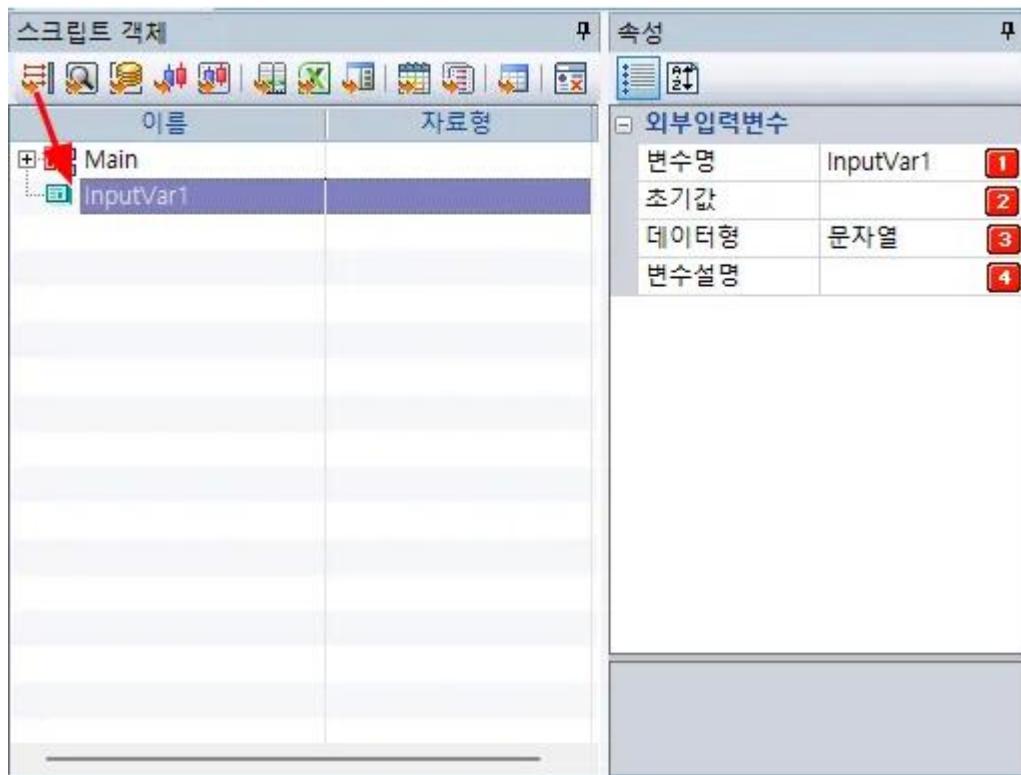
스크립트의 생성과 전략을 작성하는데 필요한 기본 Method들을 제공하는 객체로 새로운 전략 생성하면 기본으로 스크립트 객체화면에 제공되는 객체입니다.  
속성화면에 설정할 내용은 없습니다.



## 2) 외부변수

스크립트 수식 내에서 사용할 외부변수를 설정합니다.

외부변수는 복수로 추가가 가능합니다.



버튼을 클릭하면 스크립트 객체화면에 외부변수가 추가됩니다.  
속성화면에서 변수명, 초기값, 데이터형, 설명 지정해 주시면 됩니다.

**1** 변수명

변수명을 지정합니다.

**2** 초기값

해당 변수의 초기값을 지정합니다.

**3** 데이터형

해당 변수의 숫자형인지 문자형인지 지정합니다.

**4** 변수설명

간단한 설명멘트를 입력합니다.

생략이 가능합니다.

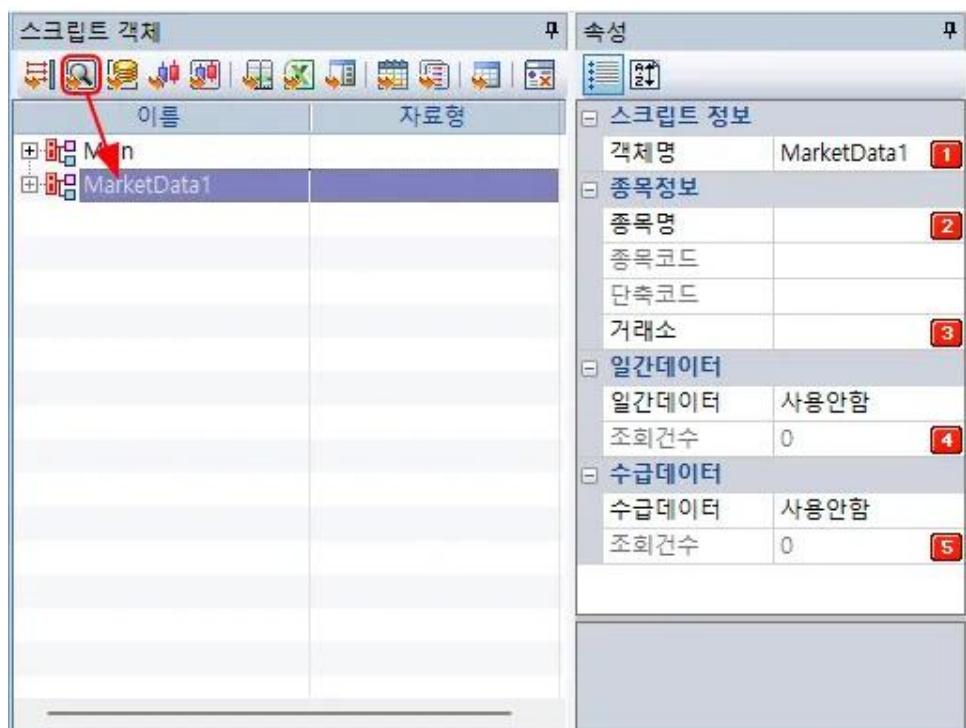
### 3) 종목객체

종목의 데이터가 제공되는 객체입니다.

시세와 호가뿐만 아니라 일간 과거데이터와 수급데이터도 참조할 수 있습니다.

종목객체는 복수로 추가가 가능하고 객체명도 임의로 부여할 수 있습니다.

스크립트 객체에 추가 후 속성화면에서 종목을 선택하고 일간데이터 사용여부를 설정해야 합니다.



버튼을 클릭하면 스크립트 객체화면에 종목객체가 추가됩니다.  
속성화면에서 종목과 거래소를 지정하고 과거 일간데이터나, 수급데이터를 사용할 경우

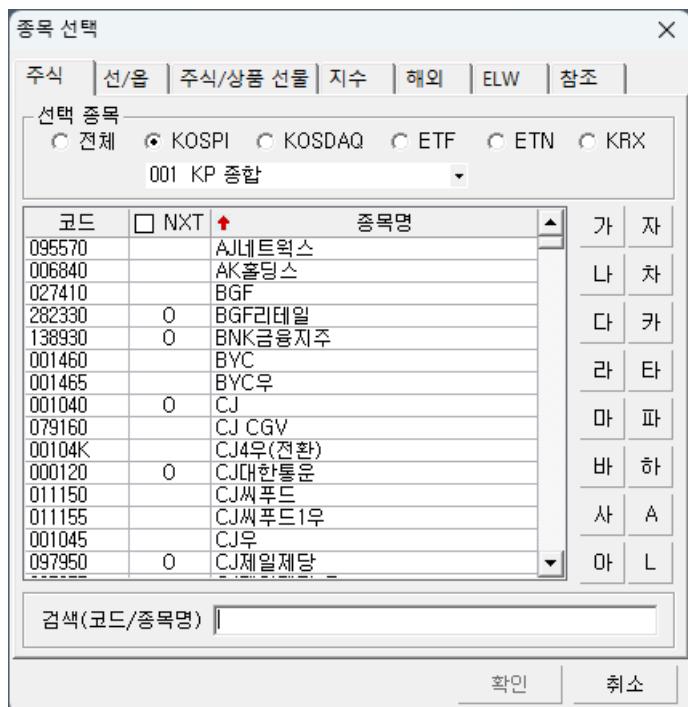
사용여부를 지정하고 갯수 지정하면 됩니다.

**1** 객체명

객체명을 지정합니다.

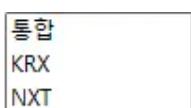
**2** 종목명

클릭하면 종목선택화면이 나타나며 종목을 지정할 수 있습니다.



**3** 거래소

시세거래소를 선택할 수 있습니다.



**4** 일간데이터

일간데이터가 필요한 경우 사용으로 설정하고 데이터 갯수를 지정하시면 됩니다.

1개에서 300개 까지 지정할 수 있습니다.

**5** 수급데이터

수급데이터가 필요한 경우 사용으로 설정하고 데이터 갯수를 지정하시면 됩니다.

1개에서 300개 까지 지정할 수 있습니다.

※수급데이터는 주식만 제공됩니다. 파생상품의 경우 참조액체에서 별도로 제공됩니다.

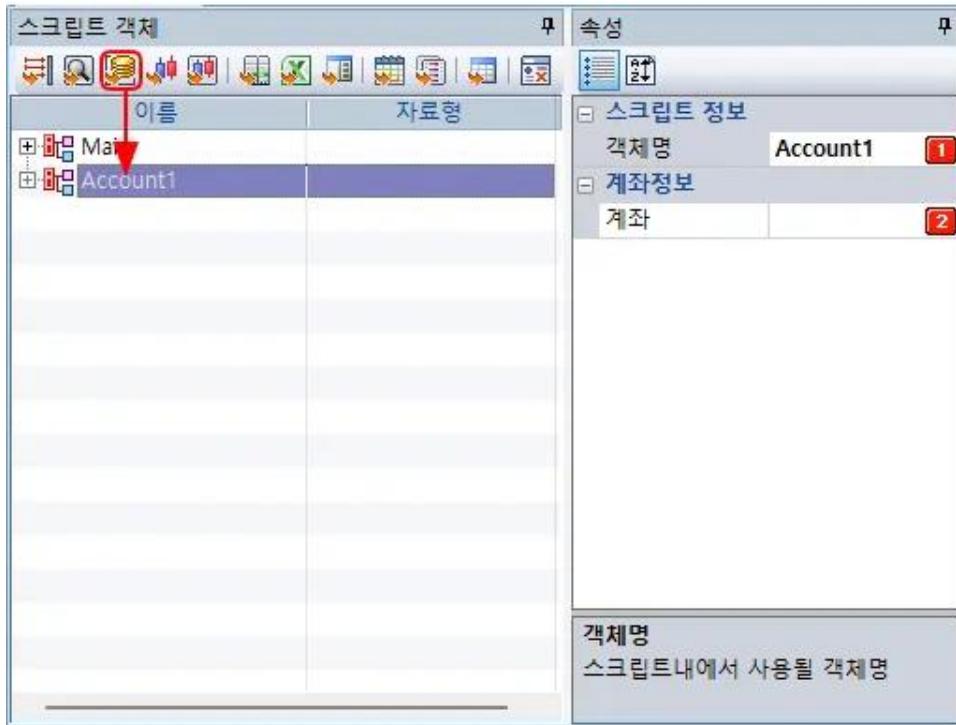
※종목액체는 수식 안에서 Main.ReqMarketData함수로 요청도 가능합니다.

전략 내용에 따라 스크립트 실행 중에 함수로 수식 내에서 호출해서 사용도 가능합니다.

#### 4) 계좌객체

지정한 계좌 정보가 제공되는 객체입니다.

계좌객체는 복수로 추가가 가능하고 객체명도 임의로 부여할 수 있습니다.



버튼을 눌러 객체리스트에 추가하고 속성화면에서 객체명과 계좌번호를 지정합니다.

**1** 객체명

객체명을 지정합니다.

**2** 계좌

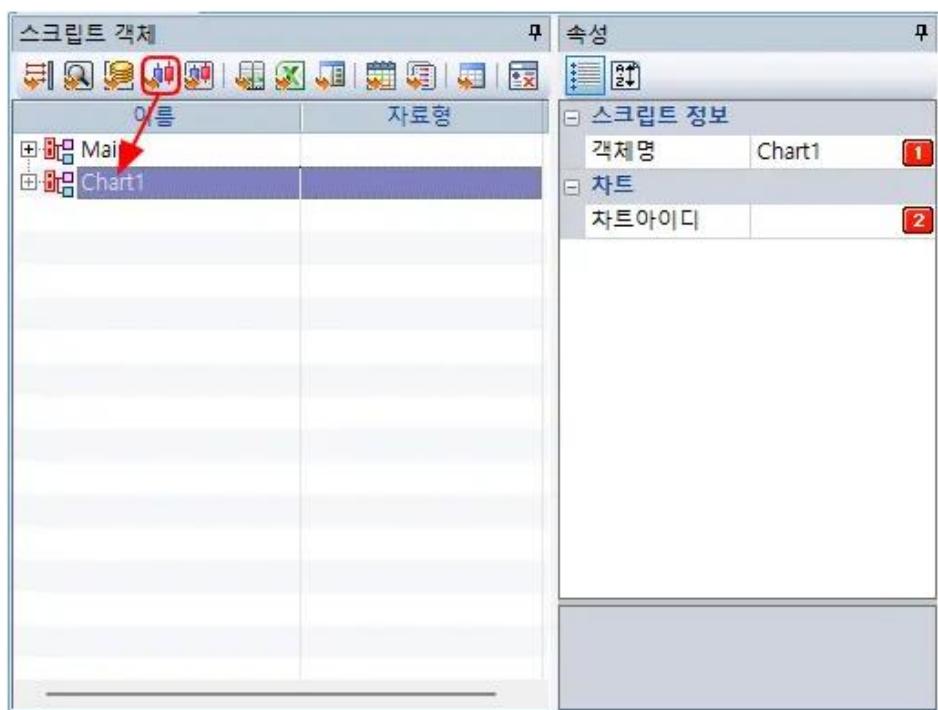
사용할 계좌번호를 지정합니다.

#### 5) 차트객체

열려진 특정 차트에서 정보(데이터, 신호, 지표값등)를 제공받는 객체입니다.

차트와 객체는 아이디를 지정하여 연결됩니다.

차트객체는 복수로 추가가 가능하고 객체명도 임의로 부여할 수 있습니다.



버튼을 클릭하면 스크립트 객체화면에 차트객체가 추가됩니다.

속성화면에서 객체명과 차트아이디를 지정하면 됩니다.

**1** 객체명

객체명을 지정합니다.

**2** 차트아이디

연결하고자 하는 차트에 지정한 아이디와 동일한 이름으로 지정합니다.

차트에 아이디 지정

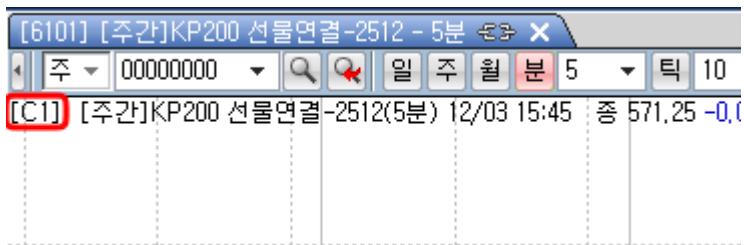
차트객체를 사용하기 위해서는 속성창의 차트아이디와 차트에 부여한 이름이 동일해야 합니다.



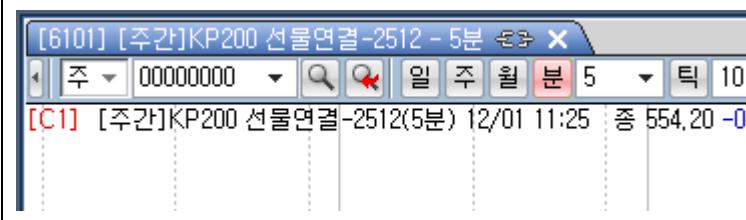
차트 하단 버튼을 클릭하시면 차트에 아이디를 부여



차트에 지정된 아이디와 차트객체 속성에서 지정한 아이디는 동일해야 합니다.  
대소문자를 구분하므로 지정시 유의하시기 바랍니다.



차트에 아이디가 지정되면 왼쪽 상단에 아이디가 표시됩니다.  
해당 차트를 사용하는 스팟식이 실행되면 빨강색으로 변경되어 사용중임을 알려주게 됩니다.



## 6) 확장차트 객체

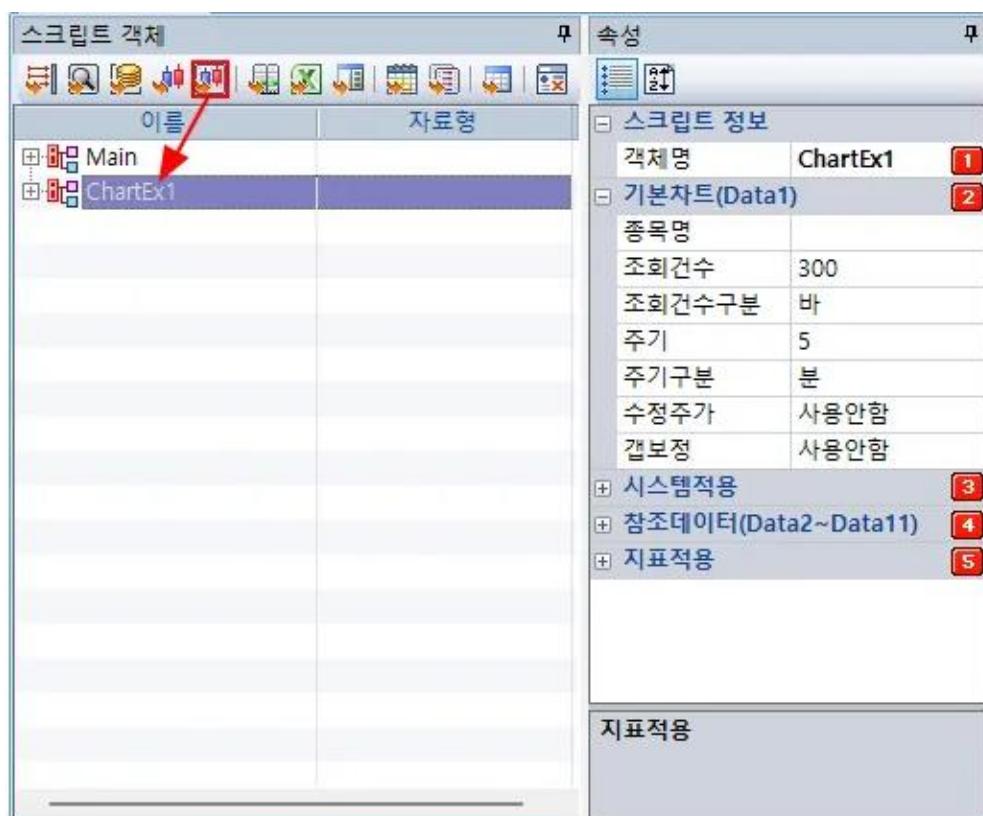
차트객체는 직접 열어 셋팅한 차트와 아이디로 연결해 사용하는 객체입니다.

확장차트 객체는 스크립트 객체화면에서 종목과 차트 기본설정을 하고

추가로 필요시 시스템, 지표, 참조데이터 등을 설정하여

스팟 실행시 설정한 내용으로 차트를 생성하여 차트를 이용하는 객체입니다.

차트객체는 복수로 추가가 가능하고 객체명도 임의로 부여할 수 있습니다.



버튼을 클릭하여 객체리스트에 추가한 후 속성에서 객체명과 생성할 차트의 종목, 해당 차트에 적용할 시스템, 지표, 참조데이터를 대해 설정합니다.

**1** 객체명

객체명을 지정합니다.

**2** 기본차트설정

종목명란을 선택하면 종목선택 화면이 나타나 종목을 선택할 수 있습니다.

종목선택 후 세부 설정항목을 설정하면 됩니다.

- 거래소 : KRX,NXT,통합 중 선택합니다.
- 시장구분 : 주간/야간/복합 중 선택합니다. 야간거래 가능한 파생상품만 선택이 가능합니다.
- 조회건수 : 조회할 데이터의 개수를 설정합니다. 최대 5000개까지 지정 가능합니다.
- 조회구분구분 : 바'이면 지정한 봉갯수로 조회를 하며, '일, 주, 월, 년'은 해당 기간을 기준으로 데이터를 조회합니다. 예를 들어 조회건수를 5로 지정하고 조회구분을 일로 설정하면 최근 5일에 해당하는 데 이터를 조회합니다. 다만, 일, 주, 월, 년으로 조회할 때 봉개수가 5000을 초과하게 되면 5000개 봉까지 만 조회합니다.
- 주기 : 차트 주기를 설정합니다.
- 주기구분 : [틱,초,분,일, 주, 월] 중 조회할 주기구분을 지정합니다.
- 수정주가 : 수정주가를 사용할 경우 사용으로 설정합니다.
- 갭보정 : 갭보정을 사용할 경우 사용으로 설정합니다.

**③ 시스템적용 설정**

시스템 적용	
사용	사용
시스템명	RSI
시스템 설정	
변수	
Period	14
LPercent	30
SPercent	70
피라미딩	허용안함
진입설정	주문수량 10000, 금
비율/수량	동일수량(계약) 진입
거래수량	1
강제청산	
손절매	
최대수익대비하락	
목표수익	
최소가격변화	
당일청산	02:45:00 PM
청산시점	조건만족즉시

시스템을 적용할 경우 사용여부를 사용으로 설정한 후 시스템명을 클릭하면 시스템 선택화면이 나타납니다.

적용할 시스템을 선택하면 시스템 트레이딩 설정창이 나타나며

강제청산이나 피라미딩 등 제반설정 후에 확인을 누르시면 됩니다.

적용된 시스템의 설정을 변경하실 때는 시스템 설정란을 클릭하면 됩니다.

이렇게 설정된 시스템의 정보는 하단의 변수, 피라미딩, 진입설정, 강제청산에 표시가 됩니다.

#### ④ 참조데이터 설정

참조데이터(Data2~Data11)	
Data2	사용
종목명	
조회건수	300
조회건수구분	바
주기	5
주기구분	분
수정주가	사용안함
갭보정	사용안함
Data3	사용안함
Data4	사용안함
Data5	사용안함
Data6	사용안함
Data7	사용안함
Data8	사용안함
Data9	사용안함
Data10	사용안함
Data11	사용안함

참조데이터는 data2~data11까지 총 10개 까지 설정하실 수 있습니다.

종목선택 후 세부 설정항목을 설정하면 됩니다.

- 시장구분 : 주간/야간/복합 중 선택합니다. 야간거래 가능한 파생상품만 선택이 가능합니다.

- 거래소 : KRX,NXT,통합 중 선택합니다.
- 조회건수 : 조회할 데이터의 개수를 설정합니다. 최대 5000개까지 지정 가능합니다.
- 조회건수구분 : 바'이면 지정한 봉갯수로 조회를 하며, '일, 주, 월, 년'은 해당 기간을 기준으로 데이터를 조회합니다. 예를 들어 조회건수를 5로 지정하고 조회건수구분을 일로 설정하면 최근 5일에 해당하는 데이터를 조회합니다. 다만, 일, 주, 월, 년으로 조회할 때 봉개수가 5000을 초과하게 되면 5000개 봉까지만 조회합니다.
- 주기 : 차트 주기를 설정합니다.
- 주기구분 : [틱,초,분,일, 주, 월] 중 조회할 주기구분을 지정합니다.
- 수정주가 : 수정주가를 사용할 경우 사용으로 설정합니다.
- 캡보정 : 캡보정을 사용할 경우 사용으로 설정합니다.

## 5 지표적용 설정

지표적용	
지표1	사용
+ 지표명	Aroon
지표2	사용
+ 지표명	Band%B
+ 지표3	사용안함
+ 지표4	사용안함
+ 지표5	사용안함

지표는 총 5개 까지만 설정할 수 있습니다.

사용설정 후 지표명란을 클릭하시면 지표선택화면이 나타납니다.

확장차트 객체를 사용한 수식을 [6131] 예스스팟화면에서 실행하면

차트 항목에 보기버튼이 나타납니다. 클릭하면 간의 차트로 보실 수 있습니다.



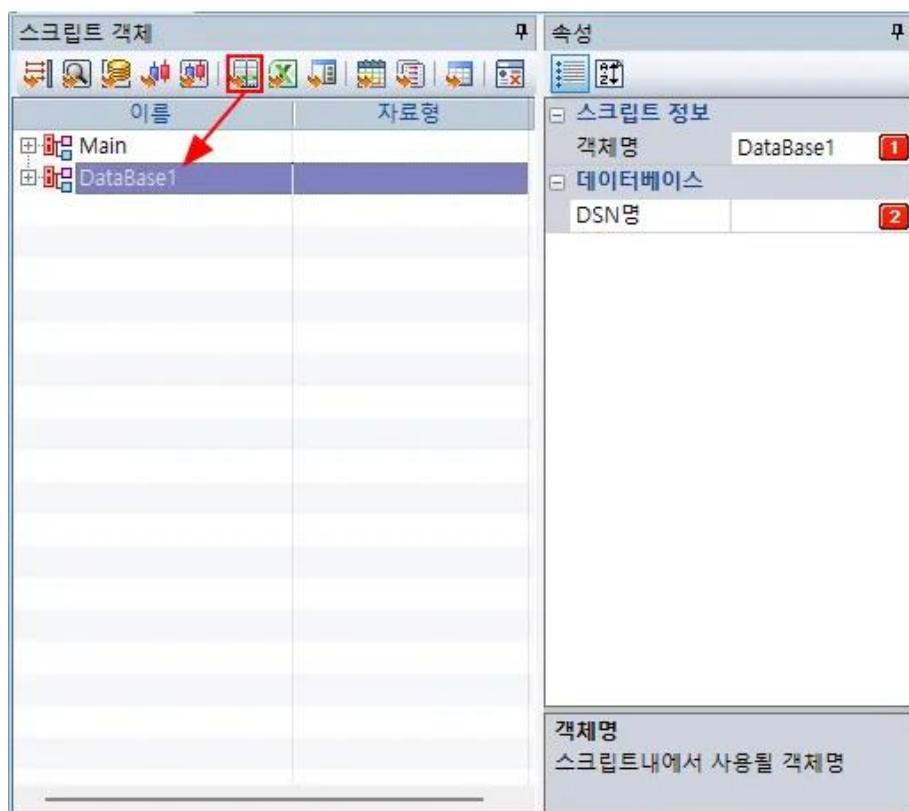
\* 확장 객체는 수식 안에서 Main.ReqChartEx 함수로 요청도 가능합니다.

전략 내용에 따라 스크립트 실행 중에 함수로 수식 내에서 확장 차트 객체를 만들어서 사용 가능합니다.

## 7) 데이터베이스 객체

ODBC 관리자를 통하여 데이터베이스에 연결할 객체입니다.

DB 객체는 복수로 추가가 가능하고 객체명도 임의로 부여할 수 있습니다.



버튼을 클릭하여 객체리스트에 추가한 후 속성화면에서 객체명과 DSN명을 지정합니다.

**1** 객체명

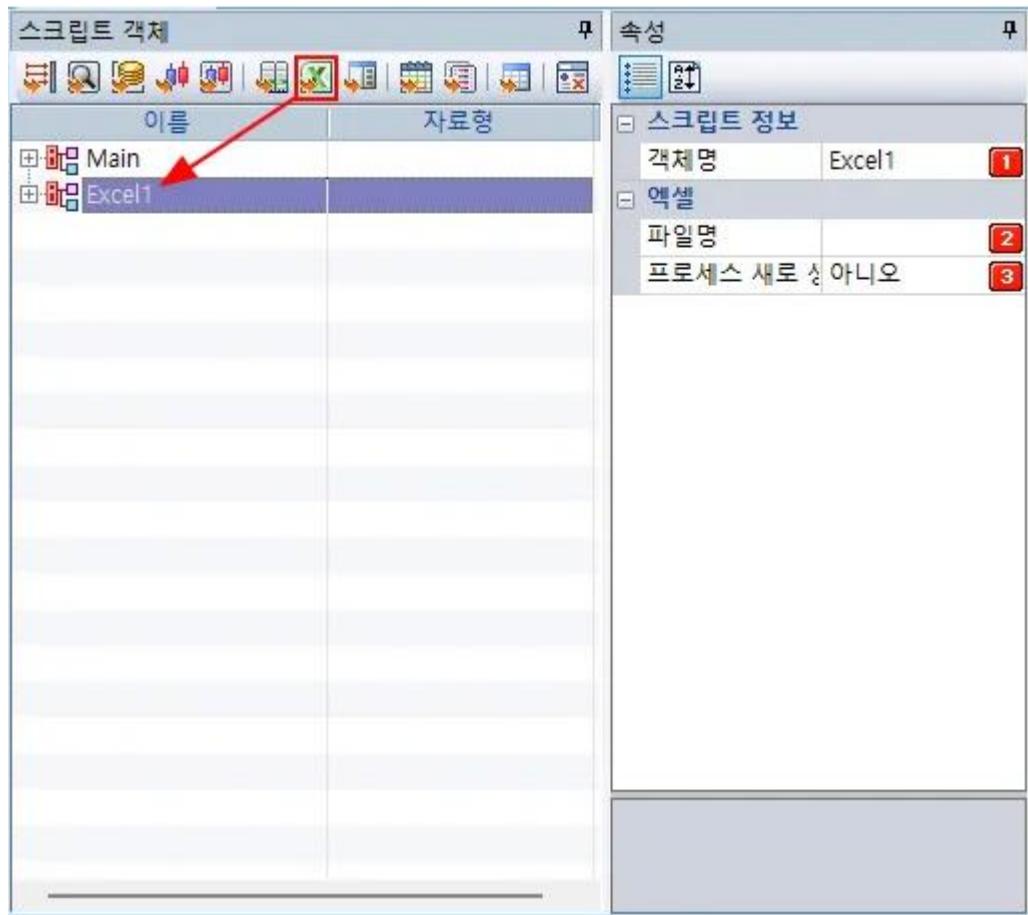
객체명을 지정합니다.

**2** DSN명

데이터베이스에서 DSN명을 지정합니다.

## 8) 엑셀데이터 객체

예스스팟과 엑셀을 연동하기 위한 객체입니다.



버튼을 클릭하면 스크립트 객체화면에 엑셀데이터객체가 추가됩니다.

속성화면에서 객체명과 엑셀파일을 지정하고 프로세스 새로 생성여부를 지정할 수 있습니다.

**1** 객체명

객체명을 지정합니다.

**2** 파일명

연동할 엑셀 파일을 지정합니다.

**3** 프로세스 새로 생성

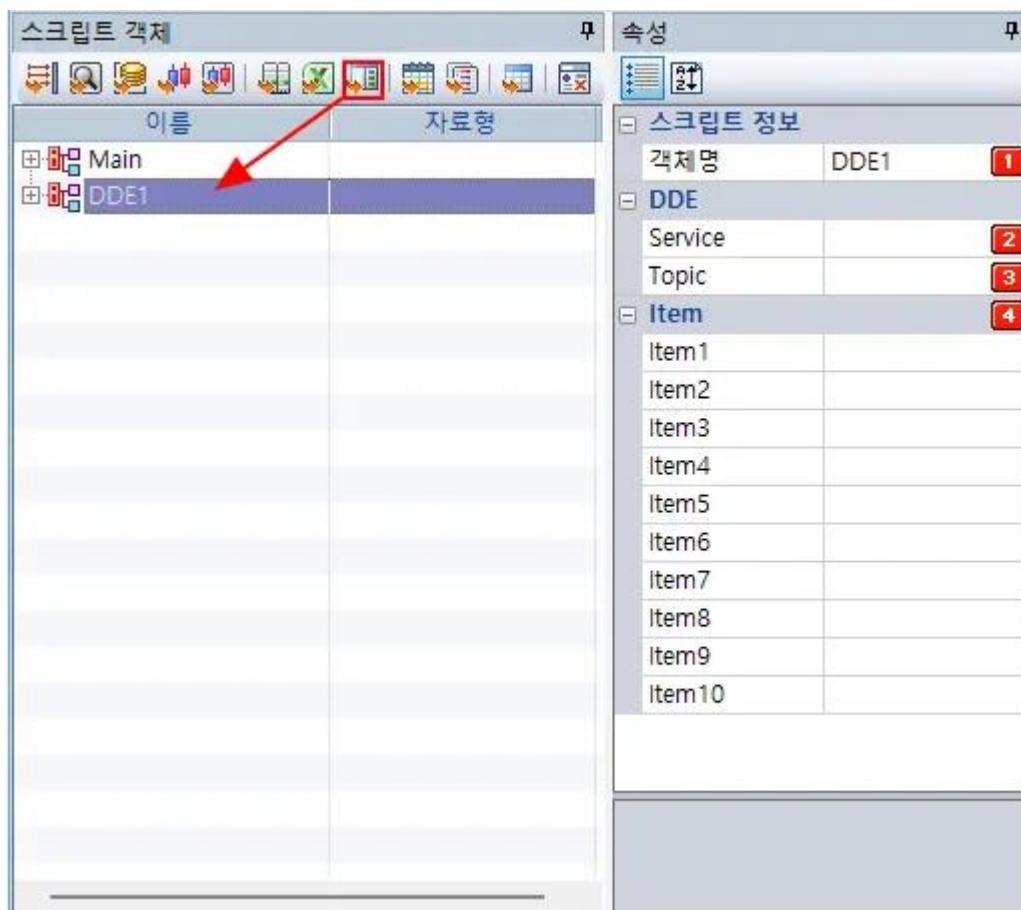
프로세스 새로 생성여부를 지정합니다.

## 9) DDE 객체

DDE 데이터를 수신받는 객체입니다.

버튼을 클릭하면 스크립트 객체화면에 DDE객체가 추가됩니다.

속성화면에서 객체명과 서비스명, 토픽명, 아이템을 설정합니다.



**1** 객체명

객체명을 지정합니다.

**2** Service

서비스명을 지정합니다.

**3** topic

토픽명을 지정합니다.

**4** item

item들을 지정합니다.



예를 들어 키움 영웅문 글로벌에서 나스닥 선물 데이터를 DDE 객체로 받는다면

DDE 서비스 화면에서 종목과 값을 설정 후

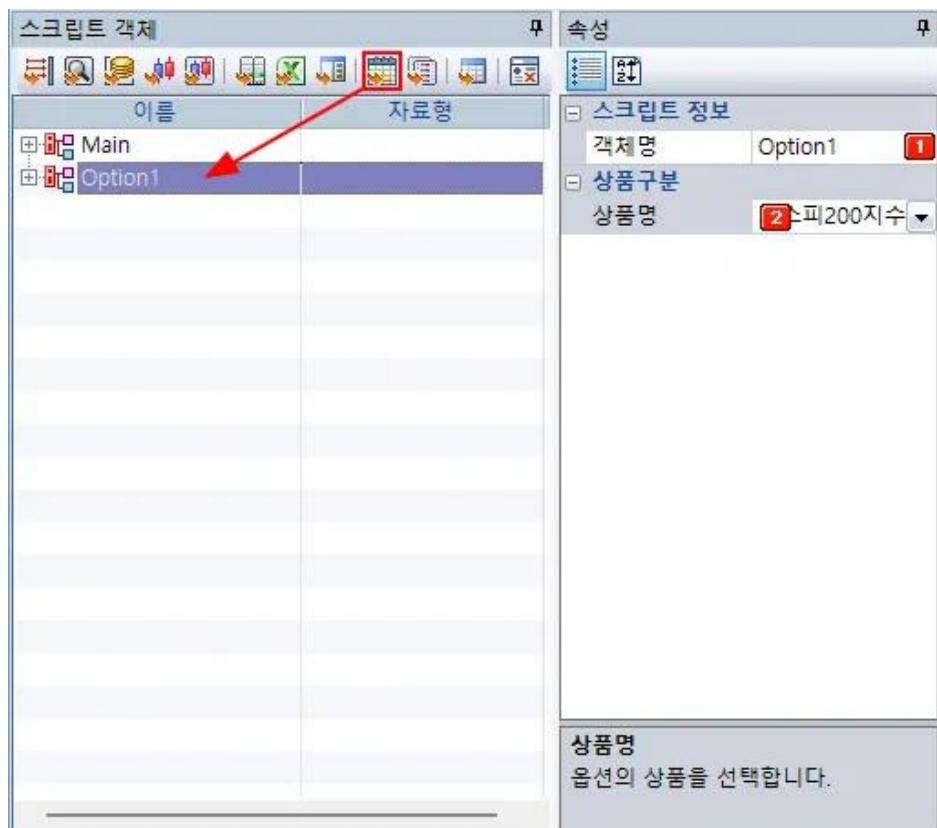
엑셀로 보내기를 해보시면 엑셀에 지정한 종목의 값이 출력됩니다.

NFRUnl Service명, NQU25가 Topic명, 1001 item입니다.

## 10) 옵션 객체

예스스팟과 엑셀을 연동하기 위한 객체입니다.

옵션데이터객체는 복수로 추가가 가능하고 객체명도 임의로 부여할 수 있습니다.



버튼을 클릭하면 스크립트 객체화면에 옵션데이터객체가 추가됩니다.

속성화면에서 객체명과 상품명에서 기초자산을 지정하면 됩니다.

**1** 객체명

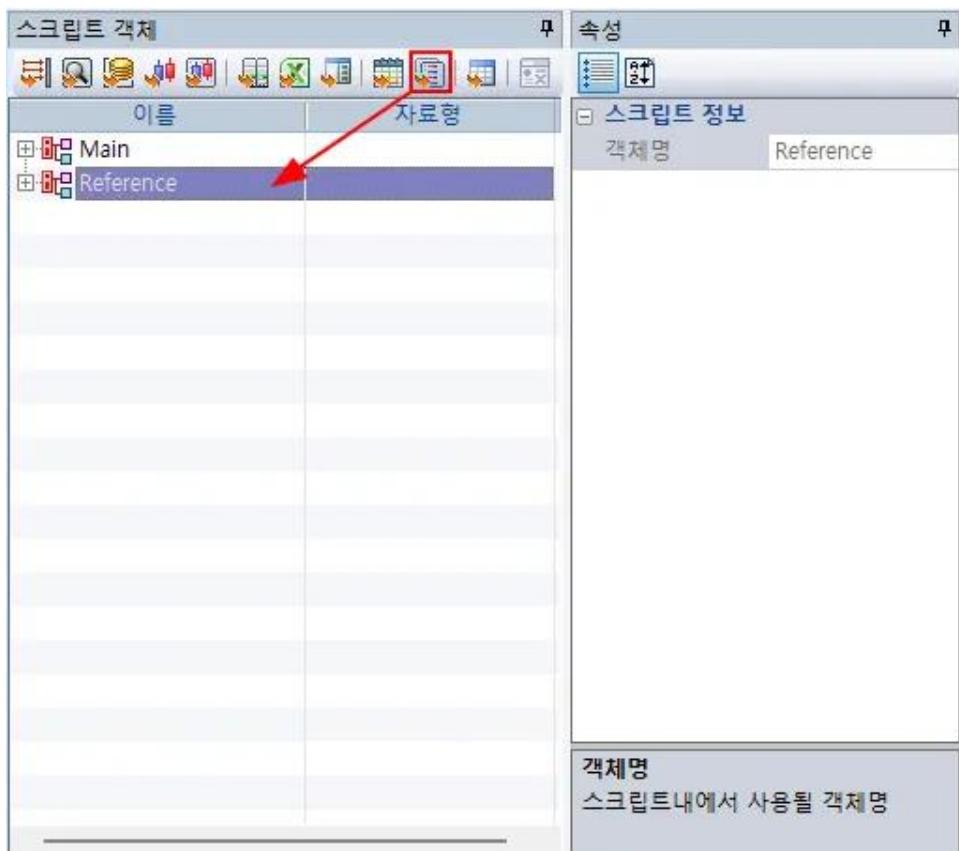
객체명을 지정합니다.

**2** 상품명

기초자산을 선택합니다.

## 11) 참조데이터 객체

시장 투자주체별 데이터등 참조데이터를 제공하는 객체입니다.



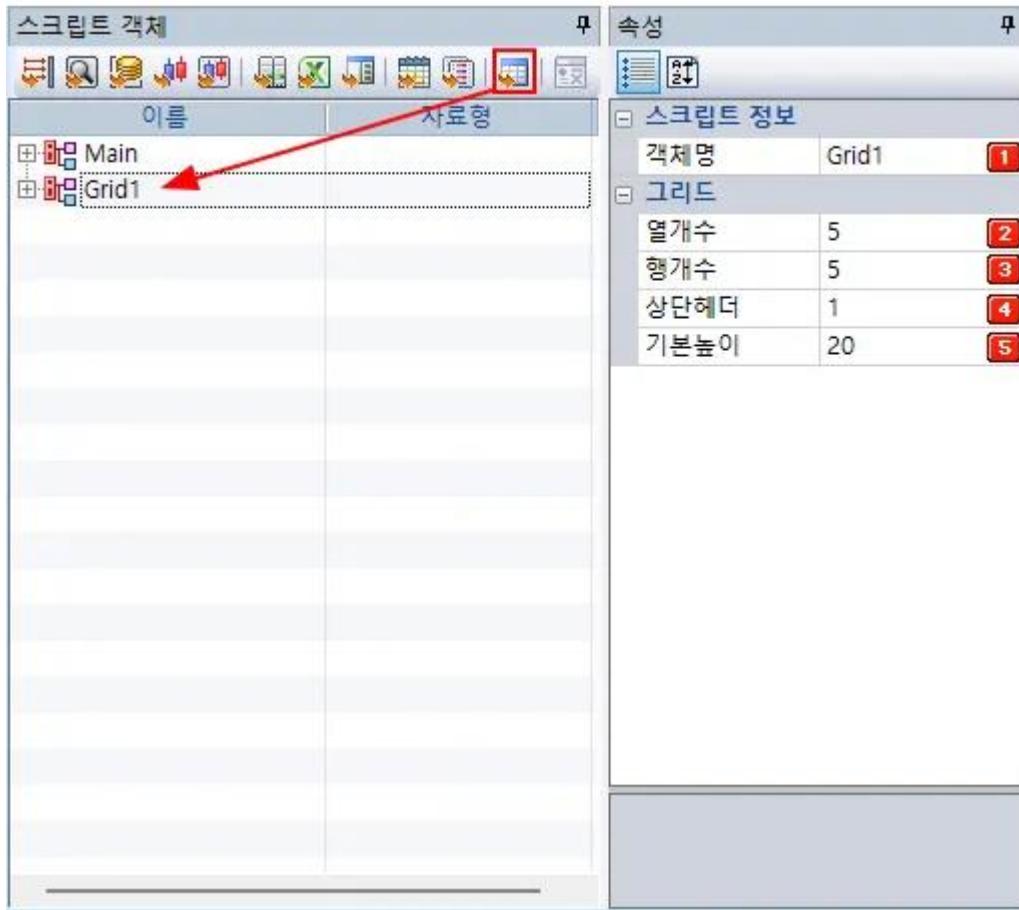
버튼을 눌러 객체리스트에 추가한 후 사용하시면 됩니다.

추가로 속성화면에서 설정할 항목은 없습니다.

## 12) 그리드 객체

값을 출력시켜 볼 수 있는 그리드 객체입니다.

그리드 객체는 복수로 추가가 가능하고 객체명도 임의로 부여할 수 있습니다.



▶ 버튼을 클릭하면 스크립트 객체화면에 그리드 객체가 추가됩니다..

**1** 객체명

객체명을 지정합니다.

**2** 열 개수

그리드의 열 개수를 지정합니다.

**3** 행 개수

그리드의 행 개수를 지정합니다.

**4** 상단 헤더

상단 헤더 사용여부를 지정합니다.

**5** 기본 높이

셀의 높이를 지정합니다.

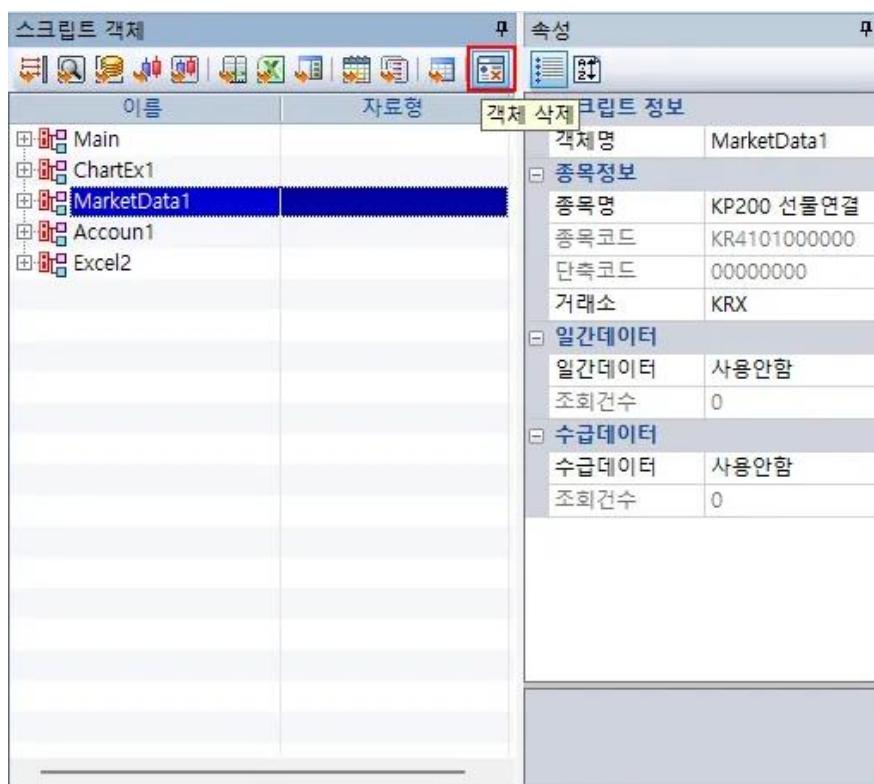
그리드 객체를 사용하는 수식을 [6131] 예스스팟화면에서 실행하면 그리드 항목에 보기버튼이 나타납니다.

클릭하면 열어서 보실 수 있습니다.



### 13) 객체삭제

스크립트 객체화면에서 객체를 선택하고 버튼을 클릭하면 해당 객체가 삭제됩니다.



※ 참조데이터 객체는 복수로 추가되지 않으므로 추가 후

삭제는  버튼을 한번 더 클릭해 주시면 됩니다

## 5. 객체, 메소드(Method), 프로퍼티(Property), 이벤트(Event) 설명

### 1) Main 객체

#### ExtractSubstring(sFullString, nIndex, sSeparator)

설명 : 전체 문자열에서 지정한 인덱스의 토큰을 추출합니다.

반환값 : 없음

매개변수 : sFullString – 문자열, 전체 문자열을 입력합니다.

nIndex – 정수, 추출할 토큰의 인덱스를 입력합니다. 시작 인덱스는 0입니다.

sSeparator – 문자열, 토큰을 분리하고 있는 문자를 입력합니다.(기본값:"|")

※ Main.ExtractSubstring("가|나|다|라|마",0,"|") → 가

※ Main.ExtractSubstring("가|나|다|라|마",3,"|") → 라

※ Main.ExtractSubstring("가|나|다|라|마",1,"|") → 나

※ Main.ExtractSubstring("가|나|다|라|마",2,"|") → 다

#### GetItemCodeInCategory(sCategoryCode, nIndex)

설명 : 지정한 업종의 지정한 순번의 종목코드를 반환합니다.

반환값 : 문자열

매개변수 : sCategoryCode – 문자열, 업종코드

nIndex – 정수, 순번(순번은 0부터 시작합니다.)

#### GetItemCodeInInterest(sInterestName, nIndex)

설명 : 지정한 관심그룹의 지정한 순번의 종목코드를 반환합니다.

반환값 : 문자열

매개변수 : sInterestName – 문자열, 관심그룹 이름

nIndex – 정수, 순번(순번은 0부터 시작합니다.)

#### GetItemCountOfCategory(sCategoryCode)

설명 : 지정한 업종에 속한 종목수를 반환합니다. (해외선물에서는 사용할 수 없습니다.)

반환값 : 정수

매개변수 : sCategoryCode – 문자열, 업종코드

KOSPI					
001	KP 종합	018	KP 건설업	201	KP200 종합
002	KP 시가총액-대	019	KP 운수창고	202	KP200 건설기계
003	KP 시가총액-중	020	KP 통신업	203	KP200 조선운송
004	KP 시가총액-소	021	KP 금융업	204	KP200 철강소재
005	KP 음식료업	022	KP 은행	205	KP200 에너지화학
006	KP 섬유,의복	024	KP 증권	206	KP200 정보통신
007	KP 종미,목재	025	KP 보험	207	KP200 금융
008	KP 화학	026	KP 서비스업	208	KP200 필수소비재
009	KP 의약품	027	KP 제조업	209	KP200 자동차소비재
010	KP 비금속광물	040	KP 배당지수	250	KSP200 선물지수
011	KP 철강,금속	050	KP50 지수	251	KSP200 선물 인버스 지수
012	KP 기계	100	KP100 지수	252	레버리지 KOSPI200 지수
013	KP 전기,전자	101	KP200 동일가중지수	260	미국달러선물지수
014	KP 의료정밀	102	KP100 동일가중지수	261	미국달러선물 인버스 지수
015	KP 운수장비	103	KP50 동일가중지수	270	커버드콜 지수
016	KP 유통업	104	KRX100 동일가중지수	271	프로텍티브포트 지수
017	KP 전기가스업			272	변동성 지수

KOSDAQ					
301	KQ 종합	343	KQ IT H/W	452	KQ 방송서비스
302	KOSDAQ 100	356	KQ 음식료,담배	453	KQ 인터넷
303	KOSDAQ MID 300	358	KQ 섬유,의류	454	KQ 디지털컨텐츠
304	KOSDAQ SMALL	362	KQ 종미,목재	455	KQ 소프트웨어
312	KQ 기타	363	KQ 출판,매체복제	456	KQ 컴퓨터서비스
315	KQ IT 종합	365	KQ 화학	457	KQ 통신장비
324	KQ 제조	366	KQ 제약	458	KQ 정보기기
326	KQ 건설	367	KQ 비금속	459	KQ 반도체
327	KQ 유통	368	KQ 금속	460	KQ IT부품
328	KQ 속박,음식	370	KQ 기계,장비	481	KQ 우량기업
329	KQ 운송	372	KQ 일반전기전자	482	KQ 벤처기업
331	KQ 금융	374	KQ 의료,정밀기기	483	KQ 중견기업
337	KQ 오락,문화	375	KQ 운송장비,부품	484	KQ 신성장기업
341	KQ통신방송서비스	377	KQ 기타제조	610	KQ KOSTAR 지수
342	KQ IT S/W, SVC	451	KQ 통신서비스	620	KQ PREMIERE 지수

KRX					
0001	KRX 100 지수	1008	KRX 필수소비재	1016	KRX 소비자유통
1001	KRX 자동차	1009	KRX 미디어통신	1017	KRX 레저엔터테인먼트
1002	KRX 반도체	1010	KRX 건설	1018	KRX 녹색산업
1003	KRX 건강	1011	KRX 비은행금융	1100	사회책임투자지수
1004	KRX 은행	1012	KRX 증권	1102	환경책임투자지수
1005	KRX 정보통신	1013	KRX 조선	1103	지배구조책임투자지수
1006	KRX 화학에너지	1014	KRX 보험		
1007	KRX 철강	1015	KRX 운송		

#### ▣ GetItemCountOfInterest(sInterestName)

설    명 : 지정한 명칭의 관심그룹에 속한 종목수를 반환합니다.

반  환  값 : 정수

매개변수 : sInterestName – 문자열, 관심그룹 이름

#### ▣ GetLastError()

설명 : 마지막으로 발생한 에러의 메시지를 반환합니다.

반환값 : 문자열

#### ▣ GetLimitedTime(nKind)

설명 : 조회나 주문 등에 시간제한이 발생했을 때 남은 시간을 1/1000초 단위로 반환합니다.

반환값 : 정수

매개변수 : 제한시간종류 (0 : 조회제한시간, 1:주문제한시간, 2 : 계좌정보조회 제한시간)

※ 종목객채 요청 – 15초당 최대 60회

※ 주문 – 15초당 최대 90회

※ 계좌 조회 – 15초당 최대 60회

#### ▣ GetMarketStatus ()

설명 : 현재 운영되고 있는 시장의 상태를 반환합니다.

반환값 : 정수 (0:정규장, 1:CME)

#### ▣ GetOrderCode(sItemCode)

설명 : 지정한 종목코드에 대한 주문용 종목코드를 리턴합니다.

반환값 : 문자열

매개변수 : strShortCode – 문자열, 종목코드(단축코드)

※ 차트에서 많이 사용하는 데이터 중 연결선물데이터나 ATM연결콜/풋옵션과 같은 데이터는 종목코드란에 표시되는데 0000000, EI736, EI741는 실제 주문코드가 아닌 해당 데이터의 일종의 데이터코드입니다. 차트에서 시스템 트레이딩 할 경우에는 현재봉의 종목으로 종목코드가 자동으로 변환되어 주문되므로 문제없지만 예스스팟에서 차트객체의 GetCode로 해당 코드를 가져와 주문함수등에 사용하게 되면 주문에 문제가 발생합니다.

이런 데이터를 사용하는 차트에서 종목코드를 가져와 사용할 때는 아래와 같이 불러온 데이터를 주문용 종목코드로 변환해서 주문함수등에 종목코드로 지정하셔야 합니다.

Main.GetOrderCode(Chart1.GetCode(1))

→ Chart1객체의 주종목의 종목코드를 주문용 종목코드로 변환

#### ▣ GetUserValue(sName)

설명 : 파일에 특정한 이름으로 저장된 값을 반환합니다.

반환값 : 문자열

매개변수 : sName – 문자열, 반환할 값의 이름을 입력합니다.

※ SetUserValue로 저장된 값은 문자열로 저장이 되므로 GetUserValue로 값을 호출하여 계산식에 사용하실 때는 저장된 값이 실수이면 parseFloat 함수를 이용하여 문자열을 실수로

변환하여 사용하셔야 하며 정수이면 parseInt로 정수로 변환하여 사용하셔야 합니다.

#### ▣ KillTimer(nEventID)

설명 : 타이머를 중지합니다.

반환값 : 없음

매개변수 : nEventID – 정수, 타이머의 ID, Settimer에서 설정한 ID를 입력합니다.

#### ▣ MessageList(sMsg)

설명 : 에스스팟 편집기의 디버깅화면으로 지정한 메시지를 출력합니다.

콤마(,)로 구분해서 출력값을 여러 개 지정할 수 있습니다.

Main.MessageList("현재가:" , MarketData2.current , "시가:" , MarketData1.open)

반환값 : 없음

매개변수 : sMsg – 문자열.

#### ▣ MessageLog(sMsg)

설명 : 에스스팟 편집기의 디버깅화면으로 지정한 메시지를 출력합니다.

반환값 : 없음

매개변수 : sMsg – 문자열.

\* MessageLog는 출력하는 값들이 복수일 경우 하나의 문자열로 만들어 출력해야 합니다

Main.MessageLog("현재가:"+MarketData1.current+"시가:"+MarketData1.open);

Main.MessageList("현재가:",MarketData1.current,"시가:",MarketData1.open);

#### ▣ OrderBuy

- OrderBuy(sAccoutnNumber, sItemCode, nCount, dPrice, nPriceKind, loanKind)

설명 : KRX 매수주문

반환값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : sAccoutnNumber- 문자열, 계좌번호, “-“를 빼고 입력합니다.

sItemCode – 문자열, 주문 네 종목의 단축코드

nCount – 정수, 주문수량

dPrice – 실수, 주문가격

nPriceKind – 정수, 가격구분

loanKind – 정수, 대출상세분류(1:유통용자매수, 2:자기용자매수)

예시 : 삼성전자 10주 시장가로 KRX 매수주문

Main.OrderBuy("1234567811","005930",10,0,1);

- Main.OrderBuy( { account:"계좌번호", exchangeKind: 거래소, code:"종목코드", count:수량, orderPrice: 주문가격, stopPrice:stop가격, priceKind:주문구분, loanKind:대출상세분류 } );

설명 : 거래소(KRX,NXT,SOR)를 지정하거나 스톱주문을 할 때는

JSON객체에 주문정보를 담아 지정해야 합니다.

반환값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : **JSON객체는 종괄호{ }안에 key:value로 이루어진 쌍들을 콤마(,) 나열해서 표현합니다.**

	Key	value
계좌번호	account	문자열, “-“는 제외
거래소	exchangeKind	정수, 1:SOR(스마트), 2:KRX, 4:NXE 생략하면 KRX
종목코드	code	문자열
주문수량	count	정수
주문가격	orderPrice	실수
스톱가격	stopPrice	실수
가격구분	priceKind	정수
대출상세분류	loanKind	정수(1:유통융자매수, 2:자기융자매수)

예시 :

```
//삼성전자 10주 시장가로 KRX
매수주문 Main.OrderBuy("1234567811","005930",10,0,1);
//삼성전자 10주 시장가로 SOR(스마트) 매수주문
Main.OrderBuy( {account:"1234567811",exchangeKind:1, code:"005930",
                count:10, orderPrice:0, priceKind:1} );
//삼성전자 10주 시장가로 KRX 매수주문
Main.OrderBuy( {account:"1234567811",exchangeKind:2, code:"005930",
                count:10, orderPrice:0, priceKind:1} );
//삼성전자 10주 시장가로 NXT 매수주문
Main.OrderBuy( {account:"1234567811",exchangeKind:4, code:"005930",
                count:10, orderPrice:0, priceKind:1} );
```

- 주문유형

priceKind	KRX	NXT	SOR(스마트)
0 : 지정가	○	○	○
1 : 시장가	○		
2 : 최유리지정가	○	○	○
3 : 지정가(IOC)	○		
4 : 시장가(IOC)	○	○	○
5 : 최유리지정가(IOC)	○		
6 : 지정가(FOK)	○		
7 : 시장가(FOK)	○	○	○
8 : 최유리지정가(FOK)	○		
9 : 조건부지정가	○		
10 : 최우선지정가	○	○	○
11 : 시간외종가(장개시 전)	○		
12 : 시간외종가(장종료 후)	○	○	
13 : 시간외단일가 매매	○		
14 : 중간가	○	○	
15 : 스톱지정가	○	○	

#### ▣ OrderCancel(sAccoutnNumber, sOrderNumber)

설명 : 취소주문

반환값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : sAccoutnNumber – 문자열, 계좌번호, “-“를 빼고 입력합니다.

sOrderNumber – 문자열, 주문번호

#### ▣ OrderReplace(sAccoutnNumber, sOrderNumber, nCount, dPrice, stopPrice)

설명 : 정정주문

반환값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : sAccoutnNumber – 문자열, 계좌번호, “-“를 빼고 입력합니다.

sOrderNumber – 문자열, 주문번호

nCount – 정수, 주문수량

dPrice – 실수, 주문가격

stopPrice – 실수, 스톱가격

#### ▣ OrderReplacePrice(sAccoutnNumber, sOrderNumber, dPrice, stopPrice)

설명 : 가격정정주문

반환 값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : sAccoutnNumber – 문자열, 계좌번호, “-“를 빼고 입력합니다.

sOrderNumber – 문자열, 주문번호

dPrice – 실수, 주문가격

stopPrice – 실수, 스탑가격

#### ▣ OrderSell

- OrderSell(sAccoutnNumber, sItemCode, nCount, dPrice, nPriceKind, loanKind, loanDate)

설명 : KRX 매도주문

반환 값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : sAccoutnNumber – 문자열, 계좌번호, “-“를 빼고 입력합니다.

sItemCode – 문자열, 주문 날 종목의 단축코드

nCount – 정수, 주문수량

dPrice – 실수, 주문가격

nPriceKind – 정수, 가격구분

loanKind – 정수, 대출상세분류(1:유통용자매수, 2:자기용자매수, 9:융자합)

loanDate – 정수, 대출날짜

예시 : 삼성전자 10주 시장가로 KRX 매도주문

Main.OrderSell("1234567811","005930",10,0,1);

- OrderSell( { account:"계좌번호", exchangeKind: 거래소, code:"종목코드", count:수량, orderPrice:주문가격, stopPrice:stop가격, priceKind:주문구분, loanKind:대출상세분류 loanDate:대출날짜 } );

설명 : 거래소(KRX,NXT,SOR)를 지정하거나 스탑주문을 할 때는

**JSON객체에 주문정보를 담아 지정해야 합니다.**

반환 값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : **JSON객체는 중괄호{ }안에 key:value로 이루어진 쌍들을 콤마(,) 나열해서 표현합니다.**

	Key	value
계좌번호	account	문자열, “-“는 제외
거래소	exchangeKind	정수, 1:SOR(스마트), 2:KRX, 4:NXE 생략하면 KRX
종목코드	code	문자열
주문수량	count	정수
주문가격	orderPrice	실수
스탑가격	stopPrice	실수
가격구분	priceKind	정수

대출상세분류	loanKind	정수(1:유통융자매수, 2:자기융자매수)
대출날짜	loanDate	정수

- 예시

```
//삼성전자 10주 시장가로 SOR(스마트) 매도주문
Main.OrderSell( {account:"1234567811",exchangeKind :1, code:"005930",
    count:10, orderPrice:0, priceKind:1} );
//삼성전자 10주 시장가로 KRX 매도주문
Main.OrderSell( {account:"1234567811",exchangeKind :2, code:"005930",
    count:10, orderPrice:0, priceKind:1} );
//삼성전자 10주 시장가로 NXT 매도주문
Main.OrderSell( {account:"1234567811",exchangeKind :4, code:"005930",
    count:10, orderPrice:0, priceKind:1} );
```

- 주문유형

priceKind	KRX	NXT	SOR(스마트)
0 : 지정가	○	○	○
1 : 시장가	○		
2 : 최유리지정가	○	○	○
3 : 지정가(IOC)	○		
4 : 시장가(IOC)	○	○	○
5 : 최유리지정가(IOC)	○		
6 : 지정가(FOK)	○		
7 : 시장가(FOK)	○	○	○
8 : 최유리지정가(FOK)	○		
9 : 조건부지정가	○		
10 : 최우선지정가	○	○	○
11 : 시간외종가(장개시 전)	○		
12 : 시간외종가(장종료 후)	○	○	
13 : 시간외단일가 매매	○		
14 : 중간가	○	○	
15 : 스톱지정가	○	○	



설명 : 스크립트의 수행을 정지합니다. 모든 이벤트들이 발생되지 않습니다.

반 환 값 : 없음

#### ▣ Play()

설 명 : 스크립트의 수행을 다시 시작합니다. 정지되었던 모든 이벤트들이 발생됩니다.

반 환 값 : 없음

#### ▣ PlaySound(sFilePath)

설 명 : 지정한 wav파일을 구동합니다.

반 환 값 : 없음

매개변수 : sFilePath – 문자열, wav 파일의 경로와 파일명

```
Main.PlaySound("C:\WSound\WStop.wav")
```

#### ▣ PrintOnFile(sFilePath, sMsg)

설 명 : 지정한 파일에 메시지를 출력합니다.

두번째 매개변수부터 가변으로 입력할 수 있습니다.

반 환 값 : 없음

매개변수 : sFilePath – 문자열, 출력할 파일경로를 지정합니다.

파일명만 입력하면 Spot → Export 디렉토리에 출력됩니다.

경로를 지정할 때 디렉토리는 \를 2개 사용해서 지정합니다.

C:\WS\에스터레이더\WS\Spot\WS\Export

sMsg – 문자열, 파일에 출력할 내용을 입력합니다.

#### ▣ RefreshAccount(sAccoutnNumber)

설 명 : 지정한 계좌의 가원장을 새로 고침합니다.

반 환 값 : 없음

매개변수 : sAccoutnNumber – 문자열, 계좌번호, ‘-’를 빼고 입력합니다.

#### ▣ RemoveIncompleteSignal(IncompleteSignal)

설 명 : GetIncompleteSignal로 얻은 미완성신호 객체를 삭제합니다.

반 환 값 : 없음

매개변수 : IncompleteSignal – 삭제할 미완성신호 객체를 입력합니다.

#### ▣ RemoveMarketData(MarketData)

설 명 : 종목객체를 삭제합니다. ReqMarketData로 생성한 종목객체만 삭제 가능합니다.

반 환 값 : 불대수

매개변수 : MarketData – 삭제할 종목객체

### RemoveObject(Object)

설명 : 객체를 삭제합니다.

수식에서 ReqMarketData, ReqChartEx 등을 이용해 동적으로 생성한 객체만 삭제할 수 있습니다.

반환값 : 정상적으로 삭제되면 1을 삭제되지 않으면 0을 반환합니다.

매개변수 : Object – 삭제할 객체

(주의 : 차트객체 삭제의 경우, 차트객체 요청 후

function Main\_OnRcvChartEx(ChartEx)에서 바로 삭제할 경우 오류가 발생 할 수 있습니다.

가급적 다른 이벤트에서 삭제를 권장합니다.)

### ReqChartEx(ReqChartItem, SystemInfo, IndicatorInfo, ReqRefItem)

설명 : 확장차트 객체 생성을 요청합니다.

반환값 : 불대수

매개변수 : ReqChartItem – 객체, 요청할 차트 종목의 기본정보, ReqChartItem 객체를 참조.

ReqChartItem(**code**, cycle, period, count, countKind, modifyPrice, dailyGap)

ReqChartItem(**{종목정보}**, cycle, period, count, countKind, modifyPrice, dailyGap)

code에 종목코드만 지정하면 주식은 KRX, 선옵은 주간장입니다.

주식종목의 거래소(통합/KRX/NXT)를 지정하거나 선

옵션 시장구분(주간/야간/복합)을 지정할 때는

code에 JSON객체로 종목정보를 담아 지정해야 합니다.

SystemInfo – 객체, 차트에 적용할 시스템 정보, SystemInfo 객체를 참조

IndicatorInfo – array, 차트에 적용할 지표 정보, IndicatorInfo 객체를 Array로 입력합니다. 기본값null

ReqRefItem – array, 차트에 적용할 타종목정보, ReqChartItem객체를 Array로 입력합니다. 기본값null

참고 : ReqChartItem 객체

ReqChartItem(**code**, cycle, period, count, countKind, modifyPrice, dailyGap)

ReqChartItem(**{종목정보}**, cycle, period, count, countKind, modifyPrice, dailyGap)

※ 코드만 지정

//기본종목셋팅(연결선을 주간장,5분 5000개, 갭보정안함, 수정주가처리 안함)

var ChartSet = new ReqChartItem("00000000",5,CHART\_PERIOD\_MINUTE,

5000,CHART\_REQCOUNT\_BAR, false, false);

//시스템 셋팅

var SysSet = new SystemInfo("Stochastics K\_D",YL\_TYPE\_NORMAL);

//지표 셋팅(지표는 복수로 지정이 가능하므로 Array에 담아 지정)

var IndSet = new Array(new IndicatorInfo("Stochastics"),new IndicatorInfo("ADX"));

//참조데이터 셋팅(ReqChartItem를 이용해 종목과 주기등지정)

//data2종목(삼성전자,5분봉 5000개,갭보정안함, 수정주가처리 안함)

```

var Data2 = new ReqChartItem("005930",5,CHART_PERIOD_MINUTE,
    5000,CHART_REQCOUNT_BAR,false,false);
//data3종목(SK하이닉스,5분봉 5000개,캡보정안함, 수정주가처리 안함)
var Data3 = new ReqChartItem("000660",5,CHART_PERIOD_MINUTE,
    5000,CHART_REQCOUNT_BAR,false,false)
//data3종목(종합주가지수,5분봉 5000개,캡보정안함, 수정주가처리 안함)
var Data4 = new ReqChartItem("001",5,CHART_PERIOD_
    MINUTE,5000,CHART_REQCOUNT_BAR,false,false) ;
//참조 데이터는 복수로 지정이 가능하므로 Array에 담아 지정
var RefSet = new Array(Data2,Data3,Data4);
//차트 셋팅 제외하고 시스템,자표, 참조데이터는 생략가능, 생략하면 null로 지정
Main.ReqChartEx(ChartSet, SysSet, IndSet, RefSet);

```

※ JSON객체로 지정

```

//기본종목셋팅({연결선물 복합장 KRX},5분 5000개, 캡보정안함, 수정주가처리 안함)
var item1 = {code:"00000000", marketKind:CHART_MARKET_CMPLX}
var ChartSet = new ReqChartItem(item1,5,CHART_PERIOD_MINUTE,
    5000,CHART_REQCOUNT_BAR,false, false);

//시스템 셋팅
var SysSet = new SystemInfo("Stochastics K_D",YL_TYPE_NORMAL);
//지표 셋팅(지표는 복수로 지정이 가능하므로 Array에 담아 지정)
var IndSet = new Array(new IndicatorInfo("Stochastics"),new IndicatorInfo("ADX"));
//참조데이터 셋팅(ReqChartItem을 이용해 종목과 주기등지정) /
//data2종목({삼성전자 통합차트},5분봉 5000개,캡보정안함, 수정주가처리 안함)
var item2 = {code:"005930", exchangeKind:1}
var Data2 = new ReqChartItem(item2,5,CHART_PERIOD_MINUTE,
    5000,CHART_REQCOUNT_BAR,false,false)
//data3종목({SK하이닉스 통합차트},5분봉 5000개,캡보정안함, 수정주가처리 안함)
var item3 = {code:"000660", exchangeKind:1}
var Data3 = new ReqChartItem(item3,5,CHART_PERIOD_MINUTE,
    5000,CHART_REQCOUNT_BAR,false,false)
//data4종목({종합주가지수 KRX차트},5분봉 5000개,캡보정안함, 수정주가처리 안함)
var item4 = {code:"001",exchangeKind:2}
var Data4 = new ReqChartItem(item4,5,CHART_PERIOD_MINUTE,
    5000,CHART_REQCOUNT_BAR,false,false)
//참조 데이터는 복수로 지정이 가능하므로 Array에 담아 지정
var RefSet = new Array(Data2,Data3,Data4);

```

```
//차트 셋팅 제외하고 시스템,자표, 참조데이터는 생략가능, 생략하면 null로 지정
Main.ReqChartEx(ChartSet, SysSet, IndSet, RefSet);
```

### ReqMarketData

- ReqMarketData(sItemCode, nDailyCount, nInvestDailyCount)

설명 : KRX시세로 종목객체 생성을 요청합니다.

반환값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : sItemCode – 문자열, 종목코드나 종목정보

nDailyCount – 정수, 일간데이터 조회건수 (0이면 일간데이터를 요청하지 않습니다.)

nInvestDailyCount – 일간수급데이터의 조회건수 (0이면 일간데이터를 요청하지 않습니다.)

예시 :

```
//삼성전자 KRX시세(일간 데이터 10개, 투자주체별 데이터 10개)
```

```
Main.ReqMarketData("005930",10,10);
```

//선물옵션은 KRX만 제공되므로 //exchangeKind 값은 무시하고 KRX 기준으로 제공됩니다.

//종목객체는 실시간 데이터를 제공하므로 주/야/복합 구분은 없습니다.

```
Main.ReqMarketData("00000000");
```

- ReqMarketData( {code : “종목코드”, exchangeKind:상수, dayCount:정수 ,refCount:정수} )

설명 : 거래소를 지정해 종목객체 생성을 요청합니다.

반환값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : 거래소를 지정해 종목객체를 요청하기 위해서는

JSON객체에 종목정보를 담아 지정해야 합니다.

```
{code : “종목코드” , exchangeKind : 상수 , dayCount : 갯수 ,refCount:갯수수}
```

JSON객체는 중괄호{ }안에 key:value로 이루어진 쌍들을 콤마(,) 나열해서 표현합니다.

	Key	Value
종목코드	code	문자열
거래소	exchangeKind	상수(1 : 통합, 2 : KRX, 4 : NXT)
일간데이터	dayCount	정수
투자주체별데이터	refCount	정수

예시 :

```
//삼성전자 통합시세(일간 데이터 10개, 투자주체별 데이터 10개)
```

```
Main.ReqMarketData({code:"005930", exchangeKind :1,dayCount:10,refCount:10});
```

//삼성전자 KRX시세(일간 데이터 10개, 투자주체별 데이터 10개)

```
Main.ReqMarketData({code:"005930", exchangeKind :2,dayCount:10,refCount:10});
```

//삼성전자 NXT시세(일간 데이터 10개, 투자주체별 데이터 10개)

```
Main.ReqMarketData({code:"005930", exchangeKind :4,dayCount:10,refCount:10});
```

//선물옵션은 KRX만 제공되므로 exchangeKind는 2(KRX)로 지정되어야 하며

//투자주체별데이터도 별도로 참조객체에서 제공되므로 종목객체에서는 지정하지 않습니다.

```
Main.ReqMarketData({code:"00000000", exchangeKind : 2,dayCount:10});
```

#### ● 요청제한

- ※ 종목객체는 15초당 최대 60회까지 요청할 수 있습니다. 15초당 60개 이하로 요청을 시간제한에 걸렸을 때 GetLimitedTime(0)로 남은 시간을 리턴 받을 수 있습니다.
- ※ 특정 종목에 대해 일간데이터와 수급데이터를 포함해 종목객체를 요청하면 조회횟수(종목데이터+일간데이터+수급데이터)는 30이 증가합니다.

#### ▣ ReqPowerSearch(sUserFunctionName)

설명 : 사용자 검색식을 이용하여 종목을 검색할 수 있습니다.

(사용자 검색식은 파워종목검색에서 만들 수 있습니다)

반환값 : 없음

매개변수 : sUserFunctionName – 문자열, 사용자검색식 이름을 지정

#### ▣ SendInterests(sGroup, altem, bAddToGroup)

설명 : 지정한 종목들을 관심그룹으로 등록합니다.

반환값 : 불대수

매개변수 : sGroup – 관심그룹명을 지정합니다.

지정한 이름이 기존에 없는 관심그룹 이름이면 새로 만들고

기존에 있는 경우, bAddToGroup를 true로 지정하면 종목을 추가하고 false면

기존 종목 리스트를 모두 삭제하고 altem에 지정한 종목으로 대체합니다.

altem – Array, 관심그룹에 보낼 종목코드 리스트를 배열로 만들어 입력합니다.

bAddToGroup – true : 기존그룹에 종목을 추가하면 true,

false : 모두 삭제하고 altem에 지정한 종목으로 대체

#### ▣ SetTimer(nEventID, nElapse)

설명 : 타이머를 실행합니다.

반환값 : 없음

매개변수 : nEventID – 정수, 타이머의 ID, OnTimer이벤트에서 구분할 수 있는 ID를 입력합니다.

nElapse – 정수, 타이머의 반복 주기를 1/1000초 단위로 설정합니다.

500을 입력하면 0.5초마다 이벤트가 발생합니다.

#### ▣ SetUserValue(sName, sValue)

설명 : 파일에 사용자가 지정한 이름으로 값을 저장합니다

지정한 값은 문자열로 저장됩니다.

반환값 : 문자열

매개변수 : sName – 문자열, 저장할 값의 이름

sValue – 문자열, 저장할 값

 OnBarAppended(ChartEx, nData)

설명 : 확장차트에서 새로운 봉의 시가가 수신되면 실행되는 이벤트입니다.

반환값 : 없음

매개변수 : ChartEx – 봉이 추가된 확장차트 객체

nData – 정수, 시가 수신된 봉의 데이터 번호(1 : data1, 2 : data2, 3 : data3, 4 : data4, ……)

 OnClose()

설명 : 예스스팟이 적용해제될 때 발생되는 이벤트입니다.

반환값 : 없음

 OnNotifyFill(NotifyContract)

설명 : 체결통보 수신 시 호출되는 이벤트입니다.

\* 프로그램에서 발생한 모든 주문에 대해서 체결통보 이벤트가 발생합니다.

반환값 : 없음

매개변수 : NotifyFill – 체결통보 객체

 OnOrderResponse(OrderResponse)

설명 : 주문응답 수신 시 실행되는 이벤트입니다.

\* 스팟전략에서 발생한 주문에 대해서만 주문응답 이벤트가 발생합니다.

반환값 : 없음

매개변수 : OrderResponse – 주문응답 객체

 OnRcvChartEx(ChartEx)

설명 : ReqChartEx함수로 요청된 확장차트 객체가 완성되었을 때 발생하는 이벤트입니다.

반환값 : 없음

매개변수 : ChartEx – 차트확장 객체

 OnRcvItemList(altemList, nCount, aValues)

설명 : 파워종목검색 결과가 있을 때 발생되는 이벤트입니다.

반환값 : 없음

매개변수 : altemList – 배열, 종목코드배열(단출코드)

nCount – 정수, 종목수(종목검색오류이면 -1을 리턴)

aValues – 배열, 결과값

#### OnRcvMarketData(MarketData)

설명 : ReqMarketData 함수로 요청된 종목객체가 완성되었을 때 호출되는 이벤트입니다.

반환값 : 없음

매개변수 : MarketData – 종목객체 객체

#### OnRiseIncompleteSignal(ChartEx, IncompleteSignal)

설명 : 확장차트 객체에서 미완성 신호가 발생하면 실행되는 이벤트입니다.

반환값 : 없음

매개변수 : ChartEx – 신호가 발생한 확장차트 객체

IncompleteSignal – 미완성신호 객체

#### OnRiseSignal(ChartEx, Signal)

설명 : 확장차트 객체에서 완성 신호가 발생하면 실행되는 이벤트입니다.

반환값 : 없음

매개변수 : ChartEx – 신호가 발생한 확장차트 객체

Signal – 완성신호 객체

#### OnStart()

설명 : 예스스팟이 처음 실행될 때 발생되는 이벤트입니다.

반환값 : 없음

#### OnTimer(nEventID)

설명 : 타이머 설정 시 발생하는 이벤트입니다.

반환값 : 없음

매개변수 : nEventID – SetTimer에서 설정한 타이머 이벤트 ID입니다.

#### OnUpdateAccount(sAccntNum, sItemCode, IUpdateID)

설명 : 계좌잔고에 변동이 있을 때 발생되는 이벤트입니다.

반환값 : 없음

매개변수 : sAccntNum – 문자열, 변동이 발생한 계좌번호.

sItemCode – 문자열, 변동이 발생한 종목코드(단축코드).

IUpdateID – 정수, 발생한 업데이트 종류

(30000: 계좌재조회, 30001 : 종목추가, 30002 : 종목삭제, 30003 : 종목변동)

#### OnUpdateMarket(sItemCode, IUpdateID)

설명 : 시세가 업데이트 될 때 발생되는 이벤트입니다.

반환값 : 없음

매개변수 : sItemCode – 문자열, 업데이트 되는 시세의 종목코드(단축코드)입니다..

IUpdateID – 정수, 업데이트 종류입니다.(20001: 시세 자동업데이트, 20002 : 호가 자동업데이트))

## 2) 종목객체

### AskTotalAmount

설명 : 매도호가 총잔량

자료형 : 정수

### AskTotalCount

설명 : 매도호가 총주문건수

자료형 : 정수

### BidTotalAmount

설명 : 매수호가 총잔량

자료형 : 정수

### BidTotalCount

설명 : 매수호가 총주문건수

자료형 : 정수

### category

설명 : 종목구분

자료형 : 정수

CategoryEx	
1	KOSPI
2	KOSDAQ
3	제3시장
4	선물
5	지수
6	옵션
7	종목옵션
8	해외종목
9	참조데이터

### categoryEx

설명 : 종목상세구분

자료형 : 정수

CategoryEx	
11	코스피 주식
12	코스닥 주식
13	코스피 ETF
14	코스닥 ETF
15	ELW
21	코스피200 선물
22	코스닥 스탠지수 선물(상폐)
23	개별주식 선물
24	미니 코스피200 선물
25	코스피200 선물 스프레드
26	개별주식선물 스프레드
27	미니 코스피200 스프레드
31	코스피200 콜옵션
32	코스피200 풋옵션
33	코스닥 스탠지수 콜옵션(상폐)
34	코스닥 스탠지수 풋옵션(상폐)
35	개별 주식 옵션
36	미니 코스피200 콜옵션
37	미니 코스피200 풋옵션
41	변동성지수 선물
42	섹터지수 선물
43	변동성지수 스프레드
44	섹터지수 스프레드
45	유로스톡스50 선물
46	유로스톡스50 스프레드
47	코스닥150 선물
48	코스닥150 스프레드
49	KRX300 선물
50	KRX300 스프레드
51	코스피지수
52	코스닥지수
53	코스닥150 콜옵션
54	코스닥150 풋옵션
61	해외데이터

71	참조데이터 중 거래가능종목
72	참조데이터 중 거래불가종목(투자주체별 데이터 같은 경우)
73	참조데이터 중 본인이 추가한 데이터
74	참조데이터 중 ATM연결 콜
75	참조데이터 중 ATM연결 풋
80	해외지수
81	해외주식

#### code

설명 : 종목코드(단축코드)

자료형 : 문자열

#### contrast

설명 : 전일대비 증감

자료형 : 실수

#### current

설명 : 현재가

자료형 : 실수

#### date

설명 : 날짜

자료형 : 정수

#### exchangeKind

설명 : 거래소구분

자료형 : 실수(1:통합, 2:KRX, 4:NXT)

#### expectedPrice

설명 : 예상체결가

자료형 : 실수

#### high

설명 : 고가

자료형 : 실수

limitDn

설명 : 하한가  
자료형 : 실수

limitUp

설명 : 상한가  
자료형 : 실수

low

설명 : 저가  
자료형 : 실수

money

설명 : 최근 체결 거래대금  
자료형 : 정수

moneyTotal

설명 : 당일 누적 거래대금  
자료형 : 정수

name

설명 : 종목명  
자료형 : 문자열

open

설명 : 시가  
자료형 : 실수

openInterest

설명 : 미결제약정  
자료형 : 정수

prevClose

설명 : 전일종가  
자료형 : 실수

#### time

설명 : 시간

자료형 : 정수

#### tradeUnit

설명 : 매매단위

자료형 : 정수

#### volume

설명 : 최근 체결 거래량

자료형 : 정수

#### volumeTotal

설명 : 당일 누적 거래량

자료형 : 정수

#### Ask(nLevel)

설명 : 매도호가

반환값 : 실수

매개변수 : nLevel – 정수, 1~10 : 매도1호가~매도10호가

#### AskAmount(nLevel)

설명 : 매도호가 잔량

반환값 : 정수

매개변수 : nLevel – 정수, 1~10 : 매도1호가~매도10호가

#### AskCount(nLevel)

설명 : 매도호가 주문건수

반환값 : 정수

매개변수 : nLevel – 정수, 1~10 : 매도1호가~매도10호가

#### Bid(nLevel)

설명 : 매수호가

반환값 : 실수

매개변수 : nLevel – 정수, 1~10 : 매수1호가~매수10호가

#### BidAmount(nLevel)

설명 : 매수호가 잔량

반환값 : 정수

매개변수 : nLevel – 정수, 1~10 : 매수1호가~매수10호가

#### BidCount(nLevel)

설명 : 매수호가 주문건수

반환값 : 정수

매개변수 : nLevel – 정수, 1~10 : 매수1호가~매수10호가

#### GetInvestorInfoByCategory(nInvestor, nAmountQty, nBuySell, nIndex)

설명 : 시장수급데이터

※ 종목객체 속성에서 수급데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 실수

매개변수 : nInvestor – 정수, 투자주체

nInvestor	주체
1	외국인
2	개인
3	금융투자
4	투신
5	은행
6	보험
7	기타금융
8	연기금
9	사모펀트
10	국가지지체
11	기타
12	외인기타

nAmountQty – 정수, 수량/금액 (0 : 수량, 1: 금액)

nBuySell – 정수, 매도/매수(1:매도, 2: 매수)

nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

#### GetInvestorInfoCount()

설명 : 시장수급데이터 수신 개수를 반환합니다.

반환값 : 정수

#### GetInvestorInfoDate(nIndex)

설명 : nIndex에 따른 시장수급데이터의 날짜를 얻습니다.

반환값 : 정수

매개변수 : 정수, (1: 1일전, 2: 2일전, 3: 3일전, …….)

#### ▣ GetPrevAsks(nIndex)

설명 : 일봉 매수잔량

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 정수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

#### ▣ GetPrevBids(nIndex)

설명 : 일봉 매도잔량

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 정수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

#### ▣ GetPrevClose(nIndex)

설명 : 일봉 종가

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 실수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

#### ▣ GetPrevCount()

설명 : 조회된 일봉의 개수,

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

※ 지정한 조회건수보다 일봉데이터의 수가 작은 종목이 있을 수 있으므로

※ 최종 조회된 일봉수를 확인할 때 사용되는 함수입니다.

반환값 : 정수

#### ▣ GetPrevDate(nIndex)

설명 : 일봉 날짜

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 정수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

#### ▣ GetPrevDnTicks(nIndex)

설명 : 일봉 하락형 체결건수

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 정수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

 GetPrevDnVol(nIndex)

설명 : 일봉 하락형 체결량

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 정수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

 GetPrevHigh(nIndex)

설명 : 일봉 고가

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 실수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

 GetPrevLow(nIndex)

설명 : 일봉 저가

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 실수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

 GetPrevMoney(nIndex)

설명 : 일봉 거래대금

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 정수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

 GetPrevOpen(nIndex)

설명 : 일봉 시가

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 실수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

 GetPrevOpenInterest(nIndex)

설명 : 일봉 미결제약정

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 정수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

 GetPrevVolume(nIndex)

설명 : 일봉 거래량

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 정수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

 GetPrevUpTicks(nIndex)

설명 : 일봉 상승형체결건수

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 정수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

 GetPrevUpVol(nIndex)

설명 : 일봉 상승형체결량

※ 종목객체 속성에서 일간데이터를 “사용”으로 설정해야 사용할 수 있습니다.

반환값 : 정수

매개변수 : nIndex – 정수, N일전, 1이상 사용가능, 정수로 입력

 GetTickSize(dPrice)

설명 : 호가단위

※ 가격(dPrice)을 지정하지 않으면 현재가로 계산을 합니다.

※ 가격(dPrice)으로 특정가격을 입력하면 해당가격의 호가단위가 리턴됩니다.

반환값 : 실수

매개변수 : dPrice – 실수, 가격

### 3) 계좌객체

 Balance

설명 : 잔고 객체

자료형 : 객체

 number

설명 : 계좌번호

자료형 : 문자열

 name

설명 : 계좌명

자료형 : 문자열

 Unfill

설명 : 미체결액체

자료형 : 객체

 GetBalanceETCinfo(nKind)

설명 : 계좌별 기타 항목 조회

반환값 : 실수

매개변수 : nKind – 정수, 예수금 관련 값의 구분

(증권사 별로 항목값이 다르므로 증권사별로 항목값 확인하시기 바랍니다.)

**-하이투자증권-**

하이투자증권			
nkind	위탁계좌	nKind	위탁계좌
0	예수금	1	위탁증거금현금+위탁증거금
2	신용보증금현금	3	신용담보금현금
4	추가담보금현금	5	단주주문
6	미수확보금	7	신용설정보증금
8	수표입금액	9	기타수표입금액
10	대용금	11	대용금과위탁증거금대용종
12	신용보증금대용	13	신용담보금대용
14	신용담보비율	15	추가담보금대용
16	권리대용금	17	출금가능금액
18	주문가능현금	19	최소주문가능금액
20	장내최대주문가능금액	21	코스닥최대주문가능금액
22	재매매대상금액	23	현금미수금
24	이자미납금	25	미상환융자금
26	기타대여금	27	융자금합계
28	대주금합계	29	담보재사용금

30	코스닥증거금현금	31	코스닥증거금대용
32	종도이용료	33	D+1추정예수금
34	D+1매도매수정산금	35	D+1매도정산금
36	D+1매수정산금	37	D+1연체변제소요금
38	D+1출금가능금액	39	D+2추정예수금
40	D+2매도매수정산금	41	D+2매도정산금
42	D+2매수정산금	43	D+2연체변제소요금
44	D+1출금가능금액	45	공모청약금
46	매도대출금액	47	비저축예수금
48	대용금현금	49	담보대출금합계
50	주문가능대용금(num)	51	계좌증거금률(var)
52	매도담보대출사용대용금	53	신용한도금액
54	신용한도잔액	55	신용최대주문가능금액
100	정산금액		

하이투자증권			
nkind	저축계좌	nKind	저축계좌
0	예수금	1	위탁증거금현금+위탁증거금
2	수표입금액	3	기타수표입금액
4	인출가능금액	5	주문가능금액
6	총주문가능금액	7	재매매대상금액
8	기타대여금	9	종도이용료
10	D+1추정예수금	11	D+1매도매수정산금
12	D+1매도정산금	13	D+1매수정산금
14	D+1연체변제소요금	15	D+1출금가능금액
16	D+2추정예수금	17	D+2매도매수정산금
18	D+2매도정산금	19	D+2매수정산금
20	D+2연체변제소요금	21	D+2출금가능금액
22	공모주청약증거금	23	비저축예수금
24	저축예수금	25	전일증거금
26	금일증거금	27	전일단주증거금
28	금일단주증거금	29	전일재매매대상금액

30	금일재매매대상금액	31	전일재매매증거금
32	금일재매매증거금	100	정산금액

하이투자증권			
nkind	선물옵션계좌	nKind	선물옵션계좌
0	예탁총액	1	예탁현금
2	예탁대용	3	선물반대매매손익
4	전일대용매도금	5	금일대용매도금
6	미수금	7	전일수표입금액
8	금일수표입금액	9	위탁증거금
10	주문증거금	11	개별종목주문증거금
12	스프레드주문증거금	13	순위형증거금증거금
14	옵션순매수대금	15	옵션매수대금
16	옵션매도대금	17	현금증거금
18	주문증거금	19	순위형증거금
20	옵션순매수대금	21	옵션매수대금
22	옵션매도대금	23	유지증거금
24	현금유지증거금	25	온션가결제총액
26	옵션가결제현금	27	추가증거금총액
28	추가증거금현금	29	주문가능총액
30	주문가능현금	31	인출가능총액
32	인출가능현금	100	정산금액

#### -NH투자증권-

NH투자증권			
nkind	주식계좌	nKind	주식계좌
-1	금일예수금		
0	D-1 일자	1	D-1 매매출금액
2	D-1 매매입금액	3	D-1 매수약정
4	D-1 매도약정	5	D 일자
6	D 예수금	7	D 매매출금액
8	D 매매입금액	9	D 40% 주문가능금액
10	D 20% 주문가능금액)	11	D 매수약정

12	D 매도약정	13	D 신용주문가능액
14	D 30% 주문가능금액	15	D 대주주문가능액
16	D 100% 주문가능금액	17	D+1 일자
18	D+1 예수금	19	D+2 일자
20	D+2 예수금	100	순자산총액

NH투자증권			
nkind	선물옵션계좌	nKind	선물옵션계좌
0	예탁총액(총평가총액)	1	인출가능금액총액 (총평가현금)
2	예탁현금 (증거금총액)	3	인출가능금액현금 (증거금현금)
4	예탁총액	5	예탁대용금액
6	익일추정예탁현금 (D+1예탁금)	7	익익일추정예탁현금 (D+2예탁금)
8	주문가능금액총액 (주문가능총액)	9	문가능금액현금 (주문가능현금)
10	옵션매도대금(옵션매도금액)	11	옵션매수대금(옵션매수금액)
12	추가증거금총액	13	위탁증거금총액
14	위탁증거금현금	15	미수금
16	전일가입금액 (전일가입금)	17	금일가입금액 (당일가입금)
100	순자산총액		

▣ GetTheNumberOfBalances()

설명 : 잔고리스트 개수

반환값 : 정수

▣ GetTheNumberOfUnfills()

설명 : 미체결리스트 개수

반환값 : 정수

▣ GetTotalAmount(nCategory, nTradeKind)

설명 : 잔고의 평가금액 합

반환값 : 실수

매개변수 : nCategory – 정수, 카테고리(0 : 전체, 1:주식, 2:선물, 3:옵션(콜+풋), 4:콜옵션, 5: 풋옵션)

nTradeKind – 정수, 매매구분(0:전체, 1:매도, 2:매수)

▣ GetTotalAvgCost(nCategory, nTradeKind)

설명 : 잔고의 평균단가 합

반환값 : 실수

매개변수 : nCategory – 정수, 카테고리(0 : 전체, 1:주식, 2:선물, 3:옵션(콜+풋), 4:콜옵션, 5: 풋옵션)  
nTradeKind – 정수, 매매구분(0:전체, 1:매도, 2:매수)

▣ GetTotalCount(nCategory, nTradeKind)

설명 : 잔고의 수량 합

반환값 : 정수

매개변수 : nCategory – 정수, 카테고리(0 : 전체, 1:주식, 2:선물, 3:옵션(콜+풋), 4:콜옵션, 5: 풋옵션)  
nTradeKind – 정수, 매매구분(0:전체, 1:매도, 2:매수)

▣ GetTotalDelta(nCategory, nTradeKind)

설명 : 잔고의 델타 합,(옵션종목에만 해당됩니다.)

반환값 : 실수

매개변수 : nCategory – 정수, 카테고리(0 : 전체, 1:주식, 2:선물, 3:옵션(콜+풋), 4:콜옵션, 5: 풋옵션)  
nTradeKind – 정수, 매매구분(0:전체, 1:매도, 2:매수)

▣ GetTotalGamma(nCategory, nTradeKind)

설명 : 잔고의 감마 합,(옵션종목에만 해당됩니다.)

반환값 : 실수

매개변수 : nCategory – 정수, 카테고리(0 : 전체, 1:주식, 2:선물, 3:옵션(콜+풋), 4:콜옵션, 5: 풋옵션)  
nTradeKind – 정수, 매매구분(0:전체, 1:매도, 2:매수)

▣ GetTotalTheta(nCategory, nTradeKind)

설명 : 잔고의 세타 합,(옵션종목에만 해당됩니다.)

반환값 : 실수

매개변수 : nCategory – 정수, 카테고리(0 : 전체, 1:주식, 2:선물, 3:옵션(콜+풋), 4:콜옵션, 5: 풋옵션)  
nTradeKind – 정수, 매매구분(0:전체, 1:매도, 2:매수)

▣ GetTotalVega(nCategory, nTradeKind)

설명 : 잔고의 베가 합,(옵션종목에만 해당됩니다.)

반환값 : 실수

매개변수 : nCategory – 정수, 카테고리(0 : 전체, 1:주식, 2:선물, 3:옵션(콜+풋), 4:콜옵션, 5: 풋옵션)  
nTradeKind – 정수, 매매구분(0:전체, 1:매도, 2:매수)

▣ OrderBuy

- OrderBuy(sItemCode, nCount, dPrice, nPriceKind, loanKind)
 

설명 : KRX 매수주문  
 반환 값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)  
 매개변수 : sAccoutnNumber- 문자열, 계좌번호, “-“를 빼고 입력합니다.  
 sItemCode – 문자열, 주문 낼 종목의 단축코드  
 nCount – 정수, 주문수량  
 dPrice – 실수, 주문가격  
 nPriceKind – 정수, 가격구분  
 loanKind – 정수, 대출상세분류(1:유통용자매수, 2:자기용자매수)  
 예시 : 삼성전자 10주 시장가로 KRX 매수주문  
`Account.OrderBuy("005930",10,0,1);`

- Main.OrderBuy( { exchangeKind: 거래소, code:"종목코드", count:수량, orderPrice:주문가격, stopPrice:stop가격, priceKind:주문구분, loanKind:대출상세분류 } );  
 설명 : 거래소(KRX,NXT,SOR)를 지정하거나 스톱주문을 할 때는  
**JSON객체에 주문정보를 담아 지정해야 합니다.**  
 매개변수 : **※ JSON객체는 중괄호{ }안에 key:value로 이루어진 쌍들을 콤마(,) 나열해서 표현합니다.**

	Key	value
거래소	exchangeKind	정수, 1:SOR(스마트), 2:KRX, 4:NXE 생략하면 KRX
종목코드	code	문자열
주문수량	count	정수
주문가격	orderPrice	실수
스톱가격	stopPrice	실수
가격구분	priceKind	정수
대출상세분류	loanKind	정수(1:유통용자매수, 2:자기용자매수)

예시 :

```
//삼성전자 10주 시장가로 SOR(스마트) 매수주문
Main.OrderBuy( {exchangeKind:1, code:"005930", count:10, orderPrice:0, priceKind:1} );

//삼성전자 10주 시장가로 KRX 매수주문
Main.OrderBuy( {exchangeKind:2, code:"005930", count:10, orderPrice:0, priceKind:1} );
//삼성전자 10주 시장가로 NXT 매수주문
Main.OrderBuy( { exchangeKind:4, code:"005930", count:10, orderPrice:0, priceKind:1} );
```

### ● 주문유형

priceKind	KRX	NXT	SOR(스마트)
0 : 지정가	○	○	○
1 : 시장가	○		
2 : 최유리지정가	○	○	○
3 : 지정가(IOC)	○		
4 : 시장가(IOC)	○	○	○
5 : 최유리지정가(IOC)	○		
6 : 지정가(FOK)	○		
7 : 시장가(FOK)	○	○	○
8 : 최유리지정가(FOK)	○		
9 : 조건부지정가	○		
10 : 최우선지정가	○	○	○
11 : 시간외종가(장개시 전)	○		
12 : 시간외종가(장종료 후)	○	○	
13 : 시간외단일가 매매	○		
14 : 중간가	○	○	
15 : 스톱지정가	○	○	

### ▣ OrderCancel(strOrderNumber)

설명 : 취소주문

반환값 : 정수(0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : strOrderNumber – 문자열, 주문 낼 종목의 단축코드

### ▣ OrderReplace(strOrderNumber, nCount, dPrice, stopPrice)

설명 : 정정주문 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

해당계좌 객체의 계좌번호로 주문되므로 계좌번호 지정하는 매개변수가 없습니다.

반환값 : 정수

매개변수 : StrOrderNumber – 문자열, 주문번호

nCount – 정수, 주문수량

dPrice – 실수, 주문가격

stopPrice – 실수, 스톱가격

### OrderReplacePrice(strOrderNumber, dPrice , stopPrice)

설명 : 가격정정주문 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

해당계좌액체의 계좌번호로 주문되므로 계좌번호 지정하는 매개변수가 없습니다.

반환값 : 정수

매개변수 : StrOrderNumber – 문자열, 주문번호

dPrice – 실수, 정정주문가격

stopPrice – 실수, 스톱가격

### OrderSell

- OrderSell(itemCode, nCount, dPrice, nPriceKind, loanKind, loanDate)

설명 : KRX 매수주문

반환값 : 정수 (0이상의 값 : 정상작동, 0 : 사용상 오류, -1: 시간제한으로 사용불가)

매개변수 : sAccoutnNumber– 문자열, 계좌번호, “-“를 빼고 입력합니다.

sItemCode – 문자열, 주문 낼 종목의 단축코드

nCount – 정수, 주문수량

dPrice – 실수, 주문가격

nPriceKind – 정수, 가격구분

loanKind – 정수, 대출상세분류(1:유통융자매수, 2:자기융자매수, 9:융자합)

loanDate – 정수, 대출날짜

예시 : 삼성전자 10주 시장가로 KRX 매도주문

```
Account.OrderSell("005930",10,0,1);
```

- Main.OrderSell( { exchangeKind: 거래소, code:"종목코드", count:수량, orderPrice:주문가격, stopPrice:stop가격, priceKind:주문구분, loanKind:대출상세분류 loanDate:대출날짜 } );

설명 : 거래소(KRX,NXT,SOR)를 지정하거나 스톱주문을 할 때는

JSON액체에 주문정보를 담아 지정해야 합니다.

※ JSON액체는 중괄호{ }안에 key:value로 이루어진 쌍들을 콤마(,) 나열해서 표현합니다.

	Key	value
거래소	exchangeKind	정수, 1:SOR(스마트), 2:KRX, 4:NXE 생략하면 KRX
종목코드	code	문자열
주문수량	count	정수
주문가격	orderPrice	실수
스톱가격	stopPrice	실수
가격구분	priceKind	정수

대출상세분류	loanKind	정수(1:유통용자매수, 2:자기용자매수)
대출날짜	loanDate	정수

예시 :

```
//삼성전자 10주 시장가로 SOR(스마트) 매도주문
Main.OrderSell( {exchangeKind :1, code:"005930", count:10, orderPrice:0, priceKind:1} );
//삼성전자 10주 시장가로 KRX 매도주문
Main.OrderSell( { exchangeKind :2, code:"005930", count:10, orderPrice:0, priceKind:1} );
//삼성전자 10주 시장가로 NXT 매도주문
Main.OrderSell( { exchangeKind :4, code:"005930", count:10, orderPrice:0, priceKind:1} );
```

#### ● 주문유형

priceKind	KRX	NXT	SOR(스마트)
0 : 지정가	○	○	○
1 : 시장가	○		
2 : 최유리지정가	○	○	○
3 : 지정가(IOC)	○		
4 : 시장가(IOC)	○	○	○
5 : 최유리지정가(IOC)	○		
6 : 지정가(FOK)	○		
7 : 시장가(FOK)	○	○	○
8 : 최유리지정가(FOK)	○		
9 : 조건부지정가	○		
10 : 최우선지정가	○	○	○
11 : 시간외종가(장개시 전)	○		
12 : 시간외종가(장종료 후)	○	○	
13 : 시간외단일가 매매	○		
14 : 중간가	○	○	
15 : 스톱지정가	○	○	



Refresh()

설명 : 계좌 가원장을 새로 고침합니다.

반환값 : 없음

▣ SetBalance(nIndex), SetBalanceIndex(nIndex),

설명 : 잔고리스트의 순서로 종목을 지정해 잔고객체를 셋팅  
(최상단 종목을 0으로 시작해 아래로 1씩 증가)

반환값 : 없음

매개변수 : nIndex – 정수, 잔고내역의 인덱스

▣ SetBalance(nIndex)

▣ SetBalance(sItemCode, nPosition)

▣ SetBalance(sItemCode, loanKind, loanDate)

설명 : 특정 종목의 잔고객체를 셋팅

\* 하나의 함수에서 매개변수의 지정에 따라 다양한 방법으로 잔고를 셋팅할 수 있습니다.

계좌리스트의 순서로 지정

종목코드와 포지션방향으로 지정

종목코드, 대출종류, 대출일로 신용매수종목을 지정

반환값 : 없음

매개변수 :

- SetBalance(nIndex)

nIndex – 정수, 잔고내역의 인덱스

- SetBalance(sItemCode, nPosition)

sItemCode – 문자열, 종목의 단축코드

nPosition – 정수, (0: 구분없음, 1:매도, 2: 매수)

- SetBalance(sItemCode, loanKind, loanDate)

sItemCode – 문자열, 종목의 단축코드

loanKind – 정수, 대출분류

(iM증권 – 1: 유통융자매수, 3: 자기융자매수, 9: 융자합)

(NH투자증권 – 0: 현금, 1: 유통융자, 3: 자기융자, 80: 대출담보)

loanDate – 정수, 대출일, 융자합일 경우 0을 입력

▣ SetBalanceIndex(sItemCode, nPosition)

설명 : 잔고리스트의 순서로 종목을 지정해 잔고객체를 셋팅

\* 최상단 종목을 0으로 시작해 아래로 1씩 증가

반환값 : 없음

매개변수 : nIndex – 정수, 잔고내역의 인덱스

▣ SetBalanceItem(sItemCode, nPosition)

설명 : 종목코드와 포지션을 지정해 잔고객체를 셋팅

반 환 값 : 없음

매개변수 : sItemCode – 문자열, 종목의 단축코드

nPosition – 정수, (0: 구분없음, 1:매도, 2: 매수)

▣ SetUnfill(nIndex)

▣ SetUnfill(sOrderNumber)

설 명 : 미체결 객체를 반환합니다.

※ 미체결리스트의 순서로 지정도 가능하고 주문번호로 지정도 가능합니다.

반 환 값 : 객체

매개변수 :

- SetUnfill(nIndex)

nIndex – 정수, 미체결 내역의 인덱스

- SetUnfill(sOrderNumber)

sOrderNumber – 문자열, 주문번호

▣ SetUnfillIndex(nIndex)

설 명 : 미체결리스트의 순서를 지정해 미체결 객체를 반환합니다.

반 환 값 : 객체

매개변수 : nIndex – 정수, 미체결 내역의 인덱스

▣ SetUnfillOrderNumber(sOrderNumber)

설 명 : 주문번호에 따른 미체결내역 객체를 반환합니다.

반 환 값 : 객체

매개변수 : sOrderNumber – 문자열, 주문번호

#### 4) 차트객체

▣ GetCode(nData)

설 명 : 지정한 차트 종목의 코드.

반 환 값 : 문자열

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

▣ GetOpen(nData, nIndex)

설 명 : 봉의 시가

반 환 값 : 실수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### GetHigh(nData, nIndex)

설명 : 봉의 고가

반환값 : 실수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### GetLow(nData, nIndex)

설명 : 봉의 저가

반환값 : 실수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### GetClose(nData, nIndex)

설명 : 봉의 종가

반환값 : 실수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### GetSDate(nData, nIndex)

설명 : 봉의 시작날짜

반환값 : 정수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### GetSTime(nData, nIndex)

설명 : 봉의 시작시간

반환값 : 정수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### GetVolume(nData, nIndex)

설명 : 봉의 거래량

반환값 : 실수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### GetOpenInterest(nData, nIndex)

설명 : 봉의 미결제약정

반환값 : 실수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,…….)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,…….)

#### GetOpenContracts()

설명 : 미청산수량을 반환합니다. 매수면 양수, 매도면 음수로 리턴됩니다.

1보다 크면 매수를 1보다 작으면 매도 포지션 상태임을 나타냅니다.

반환값 : 정수

#### GetIndicatorData(sIndicatorName, nPlotNum, nPlotIndex)

설명 : 차트에 적용된 지표식 값을 반환합니다.

반환값 : 실수

매개변수 : sIndicatorName – 문자열, 차트에 적용된 지표 이름

nPlotNum – 정수, plot번호 0부터

nPlotIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,…….)

#### GetPaintBarData(sPaintBarName, nPlotNum, nPlotIndex)

설명 : 차트의 적용된 강조식 값을 반환합니다.

반환값 : 실수

매개변수 : sIndicatorName – 문자열, 차트에 적용된 강조식 이름

nPlotNum – 정수, plot번호 0부터

nPlotIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,…….)

#### GetShowMeData(sShowMeName, nPlotNum, nPlotIndex)

설명 : 차트의 적용된 검색식 값을 반환합니다.

반환값 : 실수

매개변수 : sIndicatorName – 문자열, 차트에 적용된 검색식 이름

nPlotNum – 정수, plot번호 0부터

nPlotIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,…….)

#### GetIncompleteSignal

설명 : 마지막(현재) 봉의 미완성신호 정보를 배열로 가져옵니다.

미완성 신호가 없으면 null을 반환합니다.

객체의 사용이 완료되면 Main.RemoveIncompleteSignal()로 삭제할 수 있습니다.

※ 미완성 신호는 동시에 여러 개가 발생할 수 있고 발생된 미완성 신호에 대한 정보들이 [0]번 방부터 차례로 저장되어 제공됩니다.

※ 함수가 호출되어 사용 후에 다음 사용을 위해 적당한 시점에 객체를 삭제할 필요가 있습니다.

이전 호출시에 세개의 Buy("B1"), Buy("B2"), Buy("B3")가 동시에 만족 중이라

[0],[1],[2]번방에 미완성신호가 저장되었고 현재 호출시에 1개의 Buy("B1")가 만족 중이면 [0]번 방에는 Buy("B1")의 정보가 저장이 되고 [1], [2]번은 이전 호출시점의 정보가 남아 있게 됩니다. 그러므로 다음 사용시를 위해 호출해서 사용하고 나면 Main.RemoveIncompleteSignal()로 저장된 모든 정보를 삭제하고 다시 저장받아 사용해야 합니다.

반환값 : Array

#### ■ DrawMark(datetime, text, tradeKind) , DrawMark(marks)

설명 : 차트에 기호를 표시합니다.

반환값 : datetime값이 문자열로 리턴

매개변수 : datetime – 문자열, 기호를 표시할 차트의 X축(날짜시간),

"YYYYMMDDHHMMSSMMMM"으로 지정,

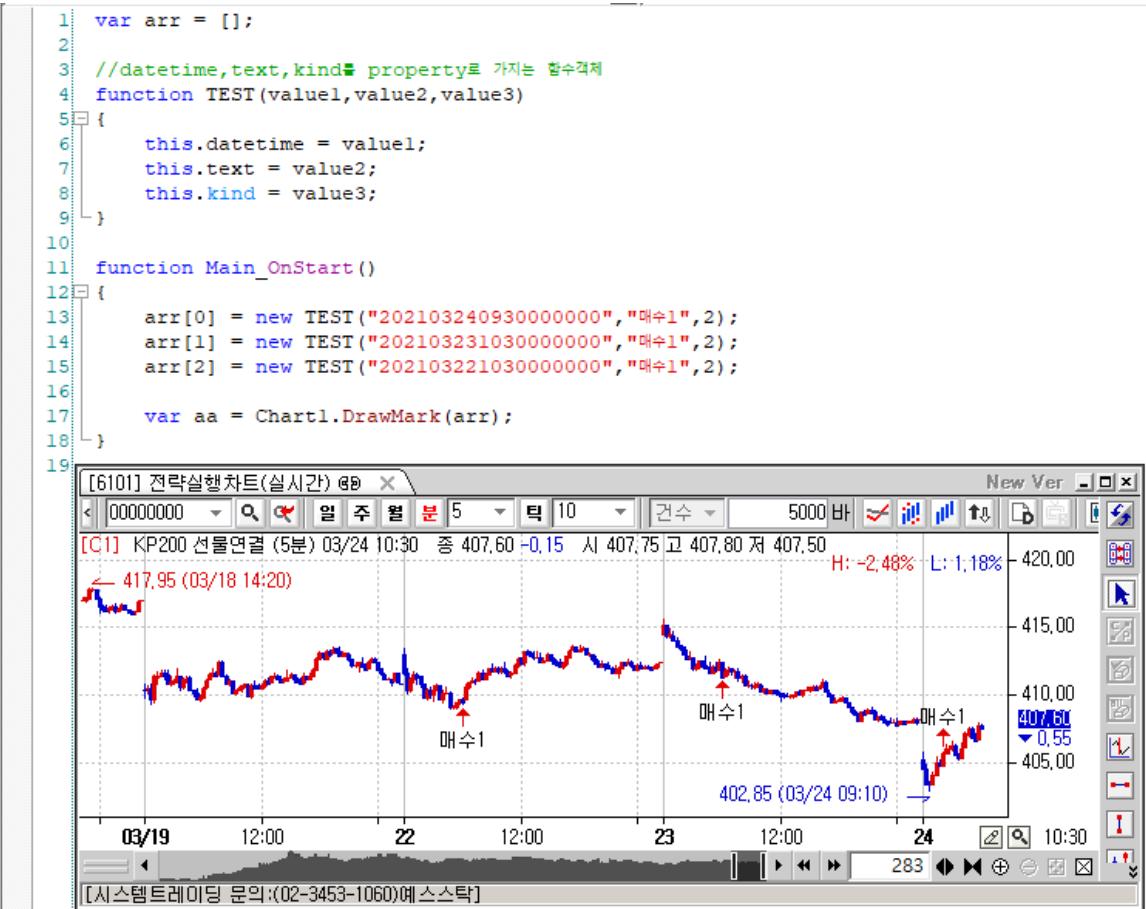
0으로 지정하면 항상 주차트의 마지막봉에 표시됨

text – 문자열, 기호와 같이 표시할 텍스트

tradeKind – 정수, 기호구분값(1:매도, 2:매수)



매개변수 : marks – 차트에 표시할 기호의 정보(datetime, text, kind를 property로 가지는 객체)에 대한 배열



#### ▣ OnBarAppended(nData)

설명 : 연결된 차트에서 새로운 봉의 시가가 수신되면 실행되는 이벤트입니다.

반환 값 : nData – 정수, 시가 수신된 봉의 데이터 번호

(1 : data1, 2 : data2, 3 : data3, 4 : data4, ……)

#### ▣ OnRiseSignal(Signal)

설명 : 연결된 차트에서 완성 신호가 발생하면 실행되는 이벤트입니다.

반환 값 : 완성신호 객체

#### ▣ OnRiseIncompleteSignal(IncompleteSignal)

설명 : 연결된 차트에서 미완성 신호가 발생하면 실행되는 이벤트입니다.

반환 값 : 미완성신호 객체

## 5) 확장차트 객체

#### ▣ GetCode(nData)

설명 : 지정한 차트 종목의 코드.  
반환값 : 문자열  
매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

#### ▣ GetOpen(nData, nIndex)

설명 : 봉의 시가  
반환값 : 실수  
매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)  
nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### ▣ GetHigh(nData, nIndex)

설명 : 봉의 고가  
반환값 : 실수  
매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)  
nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### ▣ GetLow(nData, nIndex)

설명 : 봉의 저가  
반환값 : 실수  
매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)  
nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### ▣ GetClose(nData, nIndex)

설명 : 봉의 종가  
반환값 : 실수  
매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)  
nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### ▣ GetSDate(nData, nIndex)

설명 : 봉의 시작날짜  
반환값 : 정수  
매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)  
nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### ▣ GetSTime(nData, nIndex)

설명 : 봉의 시작시간

반 환 값 : 정수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### ▣ GetVolume(nData, nIndex)

설 명 : 봉의 거래량

반 환 값 : 실수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### ▣ GetOpenInterest(nData, nIndex)

설 명 : 봉의 미결제약정

반 환 값 : 실수

매개변수 : nData – 정수, 차트 데이터 선택(1: data1, 2: data2, 3: data3, 4 : data4,……..)

nIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### ▣ GetOpenContracts()

설 명 : 미청산수량을 반환합니다. 매수면 양수, 매도면 음수로 리턴됩니다.

1보다 크면 매수를 1보다 작으면 매도 포지션 상태임을 나타냅니다.

반 환 값 : 정수

#### ▣ GetIndicatorData(sIndicatorName, nPlotNum, nPlotIndex)

설 명 : 차트에 적용된 지표식 값을 반환합니다.

반 환 값 : 실수

매개변수 : sIndicatorName – 문자열, 차트에 적용된 지표 이름

nPlotNum – 정수, plot번호 0부터

nPlotIndex – 정수, 이전값 참조(0:현재봉, 1: 1봉전, 2: 2봉전, 3: 3봉전,……..)

#### ▣ GetIncompleteSignal

설 명 : 마지막(현재) 봉의 미완성신호 정보를 배열로 가져옵니다.

미완성 신호가 없으면 null을 반환합니다.

객체의 사용이 완료되면 Main.RemoveIncompleteSignal() 로 삭제할 수 있습니다.

\* 미완성 신호는 동시에 여러 개가 발생할 수 있고 발생된 미완성 신호에 대한 정보들이 [0]번 방부터 차례로 저장되어 제공됩니다.

\* 함수가 호출되어 사용 후에 다음 사용을 위해 적당한 시점에 객체를 삭제할 필요가 있습니다.  
이전 호출시에 세개의 Buy("B1"), Buy("B2"), Buy("B3")가 동시에 만족 중이라

[0],[1],[2]번방에 미완성신호가 저장되었고 현재 호출시에 1개의 Buy("B1")가 만족 중이면 [0]번 방에는 Buy("B1")의 정보가 저장이 되고 [1], [2]번은 이전 호출시점의 정보가 남아 있게 됩니다. 그러므로 다음 사용시를 위해 호출해서 사용하고 나면 Main.RemoveIncompleteSignal()로 저장된 모든 정보를 삭제하고 다시 저장받아 사용해야 합니다.

반 환 값 : Array

## 6) 데이터베이스 객체

▣ GetFieldCount()

설 명 : SQL명령의 결과 필드의 개수를 반환합니다.

반 환 값 : 정수

▣ GetFieldValue(nIndex)

설 명 : 현재 레코드의 각 필드별 데이터를 반환합니다.

반 환 값 : 문자열

매개변수 : nIndex – 정수, 필드의 인덱스를 입력합니다.

▣ IsBOF()

설 명 : 현재 위치가 첫번째 레코드인지 여부를 반환합니다

반 환 값 : 불대수

▣ IsEOF()

설 명 : 현재 위치가 마지막 레코드인지 여부를 반환합니다.

반 환 값 : 불대수

▣ MoveFirst()

설 명 : 접근 레코드를 첫번째 레코드로 설정합니다.

반 환 값 : 없음

▣ MoveLast()

설 명 : 접근 레코드를 마지막 레코드로 설정합니다.

반 환 값 : 없음

▣ MoveNext()

설 명 : 접근 레코드를 현재 레코드의 다음 레코드로 설정합니다.

반 환 값 : 없음

### MovePrev()

설명 : 접근 레코드를 현재 레코드의 이전 레코드로 설정합니다.

반환값 : 없음

### Delete(sQuery)

설명 : SQL의 DELETE 쿼리를 데이터베이스에 전송합니다.

반환값 : 없음

매개변수 : sQuery – 문자열, 데이터베이스에 전송할 DELETE 쿼리에서 DELETE FROM 키워드를 제외한 나머지 문장을 입력합니다.

예) DELETE FROM[TableName] WHERE …쿼리를 실행하고자 한다면

DataBase1.Delete("[TableName]WHERE…")와 같이 스크립팅합니다.

### Insert(sQuery)

설명 : SQL의 INSERT 쿼리를 데이터베이스에 전송합니다.

반환값 : 없음

매개변수 : sQuery – 문자열, 데이터베이스에 전송할 INSERT 쿼리에서 INSERT INTO 키워드를 제외한 나머지 문장을 입력합니다.

예) INSERT INTO[TableName] VALUES(,,,) 쿼리를 실행하고자 한다면

DataBase1.Insert("[TableName]VALUES(,,,)")와 같이 스크립팅합니다.

### Select(sQuery, sTabTitle)

설명 : SQL의 SELECT 쿼리를 데이터베이스에 전송합니다.

반환값 : 불대수

매개변수 : sQuery – 문자열, 데이터베이스에 전송할 SELECT 명령에서 SELECT키워드를 제외한 나머지 문장을 입력합니다.

예) SELECT \* FROM [TableName] 쿼리를 실행하고자 한다면

DataBase1.Select("\* FROM [TableName]")와 같이 스크립팅합니다.

sTabTitle – 문자열, YesSpot Studio의 실행결과창에서 추가/수정할 탭의 이름을 입력합니다.

### Update(sQuery)

설명 : SQL의 UPDATE 쿼리를 데이터베이스에 전송합니다.

반환값 : 없음

매개변수 : sQuery – 문자열, 데이터베이스에 전송할 UPDATE 쿼리에서 UPDATE 키워드를 제외한 나머지 문장을 입력합니다.

예) UPDATE [TableName] SET … 쿼리를 실행하고자 한다면

DataBase!.Update("[TableName] SET …")와 같이 스크립팅합니다.

## 7) 엑셀데이터 객체

### ■ Clear(nSheetIndex, sStartCell, sEndCell)

설명 : 엑셀의 지정한 쉬트에서 지정한 셀의 영역을 지웁니다.

반환값 : 없음

매개변수 : nSheetIndex – 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

sStartCell – 문자열, 시작셀을 지정합니다. 예) “A1”

sEndCell – 문자열, 끝셀을 지정합니다. 예) “C1”

### ■ GetData(nSheetIndex, sCell)

설명 : 엑셀의 지정한 쉬트에서 지정한 셀의 값을 반환합니다.

반환값 : 문자열

매개변수 : nSheetIndex – 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

sCell – 문자열, 값을 반환할 셀을 지정합니다. 예) “A1”

### ■ SetData(nSheetIndex, sCell, aValue)

설명 : 엑셀의 지정한 쉬트에서 지정한 셀에 값을 입력합니다.

반환값 : 없음

매개변수 : nSheetIndex – 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

sCell – 문자열, 값을 입력할 시작셀을 지정합니다. 예) “A1”

aValue – 문자열, 셀에 입력할 값을 지정합니다.

### ■ GetSheetCount()

설명 : 쉬트의 개수를 반환합니다.

반환값 : 정수

### ■ GetColCount(nSheetIndex)

설명 : 지정한 쉬트의 열개수를 반환합니다.

반환값 : 정수

매개변수 : nSheetIndex – 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

### ■ GetRowCount(nSheetIndex)

설명 : 지정한 쉬트의 행 개수를 반환합니다.

반환값 : 정수

매개변수 : nSheetIndex – 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

### SetColData(nSheetIndex, sCell, aValue1)

설명 : 엑셀의 지정한 쉬트에서 지정한 셀을 시작으로 입력된 매개변수만큼 열에 값을 입력합니다.

반환 값 : 없음

매개변수 : nSheetIndex – 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

sCell – 문자열, 값을 입력할 시작셀을 지정합니다. 예) “A1”

aValue – 문자열, 셀에 입력할 값을 지정합니다. 값을 지정하는 매개변수는 가변입니다.

### SetRowData(nSheetIndex, sCell, aValue1)

설명 : 지정한 차트 종목의 코드.

반환 값 : 객체

매개변수 : nSheetIndex – 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

sCell – 문자열, 값을 입력할 시작셀을 지정합니다. 예) “A1”

aValue – 문자열, 셀에 입력할 값을 지정합니다. 값을 지정하는 매개변수는 가변입니다.

### SetArrayToCol(nSheetIndex, sCell, aValue)

설명 : 엑셀의 지정한 쉬트에서 지정한 셀을 시작으로 열에 입력된 배열객체의 값을 입력합니다.

반환 값 : 없음

매개변수 : nSheetIndex – 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

sCell – 문자열, 값을 입력할 시작셀을 지정합니다. 예) “A1”

aValue – Array, 셀에 입력할 값을 배열로 만들어 지정합니다.

### SetArrayToRow(nSheetIndex, sCell, aValue)

설명 : 엑셀의 지정한 쉬트에서 지정한 셀을 시작으로 행에 입력된 배열객체의 값을 입력합니다.

반환 값 : 없음

매개변수 : nSheetIndex – 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

sCell – 문자열, 값을 입력할 시작셀을 지정합니다. 예) “A1”

aValue – Array, 셀에 입력할 값을 배열로 만들어 지정합니다.

### SetBkColor(nSheetIndex, sCell, nColor)

설명 : 엑셀의 지정한 쉬트에서 지정한 셀의 배경색을 지정합니다.

반환 값 : 없음

매개변수 : nSheetIndex – 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

sCell – 문자열, 배경색을 지정할 시작셀을 지정합니다. 예) “A1”

nColor – 정수, 배경색을 지정합니다.

### SetAlign(nSheetIndex, sCell, nHAlign, nVAlign, nOrientation)

설명 : 엑셀의 지정한 쉬트에서 지정한 셀의 텍스트 정렬상태를 지정합니다.

반환 값 : 없음

매개변수 : nSheetIndex - 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

sCell - 문자열, 셀을 지정합니다. 예) "A1"

nHAlign - 정수, 텍스트의 가로정렬을 지정합니다

(XL\_LEFT, XL\_RIGHT, XL\_CENTER등 엑셀 관련 상수를 참고하시기 바랍니다)

nVAlign - 정수, 셀에 입력할 값을 지정합니다.

(XL\_TOP, XL\_BOTTOM, XL\_CENTER등 엑셀 관련 상수를 참고하시기 바랍니다)

nOrientation - 정수, 셀에 입력할 값을 지정합니다.

### SetFocus(bFocus)

설명 : 엑셀의 셀이 변경되었을 때 포커스 시킬지 여부를 설정합니다.

반환 값 : 없음

매개변수 : bFocus - 불대수, 셀이 변경되었을 때 포커스 시키지 여부를 설정합니다.

### SetFont(nSheetIndex, sCell, nFontSize, sFontName, nColor, bItalic, bBold, nUnderLine)

설명 : 엑셀의 지정한 쉬트에서 지정한 셀의 글꼴을 지정합니다.

반환 값 : 없음

매개변수 : nSheetIndex - 정수, 쉬트번호를 지정합니다. 첫번째 쉬트는 1부터 시작합니다.

sCell - 문자열, 글꼴을 지정할 시작셀을 지정합니다. 예) "A1"

nFontSize - 정수, 글꼴의 크기를 지정합니다.

sFontName - 문자열, 글꼴 이름을 지정합니다.

nColor - 정수, 글꼴 색깔을 지정합니다.

bItalic - 불대수, 글꼴을 이탈릭체로 표시합니다.

bBold - 불대수, 글꼴을 진하게 표시합니다.

nUnderLine - 정수, 글꼴에 밑줄을 표시합니다.(1: 밑줄해제, 2:한줄밑줄, 3: 두줄밑줄)

### Save()

설명 : 수정된 엑셀 문서를 저장합니다.

반환 값 : 없음

## 8) DDE 객체

### service

설명 : 서비스명

자료형 : 문자열

#### topic

설명 : 토픽명

자료형 : 문자열

#### Request(sItem)

설명 : 지정된 아이템의 데이터를 요청합니다.

반환값 : 없음

매개변수 : sItem – 아이템이름

#### OnRcvItem(sItem, sData)

설명 : 수신된 DDE 아이템이 있을 때 발생하는 이벤트입니다.

반환값 : sItem – 문자열, 아이템이름

sData – 문자열, 수신된 데이터

### 9) Option 객체

#### lowersATM

설명 : ATM 행사가보다 작은 행사가의 개수

자료형 : 정수

#### uppersATM

설명 : ATM 행사가보다 큰 행사가의 개수

자료형 : 정수

#### RiskFreeRate

설명 : 무위험이자율(CD금리)

자료형 : 실수

#### Volatility

설명 : 변동성

자료형 : 실수

#### GetATMCallRecent(nLevel ,nNext)

설명 : ATM 콜옵션 최근월물의 종목코드를 가져옵니다.

반환값 : 문자열

매개변수 : nLevel – 정수, ATM +-단계(0 : ATM. 양수 : +단계, 음수 : -단계)

nNext – 정수, (0: 근월물, 1:차월물), 생략가능(생략하면 근월물)

#### GetATMPutRecent(nLevel,nNext)

설명 : ATM 풋옵션 최근월물의 종목코드를 가져옵니다.

반환값 : 문자열

매개변수 : nLevel – 정수, ATM +-단계(0 : ATM. 양수 : +단계, 음수 : -단계)

nNext – 정수, (0: 근월물, 1:차월물), 생략가능(생략하면 근월물)

#### GetCodeByExercisePrice(nCallPut, dExercisePrice)

설명 : 지정한 행사가격에 해당하는 종목코드를 반환합니다.

반환값 : 문자열

매개변수 : nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

dExercisePrice – 실수, 행사가격

#### GetAsk

설명 : 지정한 콜/풋 옵션의 매도호가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetAsk(nCallPut, nATMLevel, nAskLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 : +단계, 음수 : -단계)

nAskLevel – 정수, 1~5 : 매도1호가~매도5호가

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetAsk(sItemCode, nAskLevel)

sItemCode – 문자열, 종목코드(단축코드)

nAskLevel – 정수, 1~5 : 매도1호가~매도5호가

#### GetAskAmount

설명 : 지정한 콜/풋 옵션의 매도호가 잔량

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 정수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetAskAmount(nCallPut, nATMLevel, nAskLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

nAskLevel – 정수, 1~5 : 매도1호가~매도5호가

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetAskAmount(sItemCode, nAskLevel)

sItemCode – 문자열, 종목코드(단축코드)

nAskLevel – 정수, 1~5 : 매도1호가~매도5호가

#### GetAskCount

설명 : 지정한 콜/풋 옵션의 매도호가 주문건수

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 정수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetAskCount(nCallPut, nATMLevel, nAskLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

nAskLevel – 정수, 1~5 : 매도1호가~매도5호가

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetAskCount(sItemCode, nAskLevel)

sItemCode – 문자열, 종목코드(단축코드)

nAskLevel – 정수, 1~5 : 매도1호가~매도5호가

#### GetAskTotalAmount

설명 : 지정한 콜/풋 옵션의 매도호가 총잔량

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 정수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetAskTotalAmount(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetAskTotalAmount(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

### GetAskTotalCount

설명 : 지정한 콜/풋 옵션의 매도호가 총주문건수

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 정수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetAskTotalCount(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetAskTotalCount(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

### GetBid

설명 : 지정한 콜/풋 옵션의 매수호가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetBid(nCallPut, nATMLevel, nBidLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

nBidLevel – 정수, 1~10 : 매수1호가~매수5호가

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetBid(sItemCode, nBidLevel)

sItemCode – 문자열, 종목코드(단축코드)

nBidLevel – 정수, 1~5 : 매수1호가~매수5호가

### GetBidAmount

설명 : 지정한 콜/풋 옵션의 매수호가 잔량

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 정수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetBidAmount(nCallPut, nATMLevel, nBidLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 : +단계, 음수 : -단계)

nBidLevel – 정수, 1~5 : 매수1호가~매수5호가

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetBidAmount(sItemCode, nBidLevel)

sItemCode – 문자열, 종목코드(단축코드)

nBidLevel – 정수, 1~5 : 매수1호가~매수5호가

#### ▣ GetBidCount

설명 : 지정한 콜/풋 옵션의 매수호가 건수

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 정수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetBidCount(nCallPut, nATMLevel, nBidLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 : +단계, 음수 : -단계)

nBidLevel – 정수, 1~5 : 매수1호가~매수5호가

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetBidCount(sItemCode, nBidLevel)

sItemCode – 문자열, 종목코드(단축코드)

nBidLevel – 정수, 1~5 : 매수1호가~매수5호가

#### ▣ GetBidTotalAmount

설명 : 지정한 콜/풋 옵션의 매수호가 총잔량

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 정수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetBidTotalAmount(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 : +단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetBidTotalAmount(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

### GetBidTotalCount

설명 : 지정한 콜/풋 옵션의 매수호가 총주문건수

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 정수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

```
GetBidTotalCount(nCallPut, nATMLevel)
```

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

```
GetBidTotalCount(sItemCode)
```

sItemCode – 문자열, 종목코드(단축코드)

### GetCurrent

설명 : 지정한 콜/풋 옵션의 현재가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

```
GetCurrent(nCallPut, nATMLevel)
```

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

```
GetCurrent(sItemCode)
```

sItemCode – 문자열, 종목코드(단축코드)

### GetHigh

설명 : 지정한 콜/풋 옵션의 당일고가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

```
GetHigh(nCallPut, nATMLevel)
```

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetHigh(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetLow

설 명 : 지정한 콜/풋 옵션의 당일저가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반 환 값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetLow(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetLow(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetOpen

설 명 : 지정한 콜/풋 옵션의 당일시가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반 환 값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetOpen(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetOpen(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetVolume

설 명 : 지정한 콜/풋 옵션의 당일거래량

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반 환 값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetVolume(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetVolume(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetOpenInterest

설명 : 지정한 콜/풋 옵션의 당일 미결제약정

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetOpenInterest(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetOpenInterest(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetExercisePrice

설명 : 지정한 콜/풋 옵션의 행사가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetExercisePrice (nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetExercisePrice (sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetExpectedPrice

설명 : 지정한 콜/풋 옵션의 예상체결가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetExpectedPrice(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetExpectedPrice(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetBasePrice

설명 : 지정한 콜/풋 옵션의 기준가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetBasePrice(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetBasePrice(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetLimitDn

설명 : 지정한 콜/풋 옵션의 하한가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetLimitDn(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetLimitDn(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetLimitUp

설명 : 지정한 콜/풋 옵션의 상한가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetLimitUp(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetLimitUp(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetDelta

설명 : 지정한 콜/풋 옵션의 델타

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetDelta(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetDelta(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetGamma

설명 : 지정한 콜/풋 옵션의 감마

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetGamma(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetGamma(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetTheta

설명 : 지정한 콜/풋 옵션의 세타

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetTheta(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetTheta(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetVega

설명 : 지정한 콜/풋 옵션의 베가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetVega(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetVega(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetDisparateRatio

설명 : 지정한 콜/풋 옵션의 괴리를

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetDisparateRatio(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetDisparateRatio(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetImpliedVolatility

설명 : 지정한 콜/풋 옵션의 내재변동성

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetImpliedVolatility(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetImpliedVolatility(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetIntrinsicValue

설명 : 지정한 콜/풋 옵션의 내재가치

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetIntrinsicValue(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 :+단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetIntrinsicValue(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetRemainDays

설명 : 지정한 콜/풋 옵션의 잔존일수

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetRemainDays(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 : +단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetRemainDays(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetTimeValue

설명 : 지정한 콜/풋 옵션의 시간가치

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetTimeValue(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 : +단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetTimeValue(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetTheoryPrice

설명 : 지정한 콜/풋 옵션의 이론가

ATM단계로 지정해 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수도 있습니다.

반환값 : 실수

매개변수1 : ATM단계로 종목을 지정해 값을 리턴하는 경우

GetTheoryPrice(nCallPut, nATMLevel)

nCallPut – 정수, 콜/풋 구분

(0 : 최근월물 콜, 1: 최근월물 풋, 2: 차근월물 콜, 3 : 차근월물 풋)

nATMLevel – 정수, ATM +-단계(0 : ATM. 양수 : +단계, 음수 : -단계)

매개변수2 : 종목코드로 종목을 지정해 값을 리턴하는 경우

GetTheoryPrice(sItemCode)

sItemCode – 문자열, 종목코드(단축코드)

※※ 아래는 현재 옵션객체 함수목록에는 제외되었지만 식 작성 시에는 사용할 수 있는 함수입니다.

옵션객체의 함수 하나로 ATM단계로 지정해서 값을 리턴받을 수도 있고 종목코드로 지정해 값을 리턴받을 수 있게 변경이 되어 기존 함수명에서 ByCode가 붙은 함수는 목록에서 제외되었습니다.

#### ▣ GetAskByCode(sItemCode, nAskLevel)

설명 : 종목코드(단축코드)로 지정한 옵션의 매도호가

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

nAskLevel – 정수, 1~5 : 매도1호가~매도5호가

#### ▣ GetAskAmountByCode(sItemCode, nAskLevel)

설명 : 종목코드(단축코드)로 지정한 옵션의 매도호가 잔량

반환값 : 정수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

nAskLevel – 정수, 1~5 : 매도1호가~매도5호가

#### ▣ GetAskCountByCode(sItemCode, nAskLevel)

설명 : 종목코드(단축코드)로 지정한 옵션의 매도호가 건수

반환값 : 정수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

nAskLevel – 정수, 1~5 : 매도1호가~매도5호가

#### ▣ GetAskTotalAmountByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션의 매도호가 총잔량

반환값 : 정수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

#### ▣ GetAskTotalCountByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션의 매도호가 총주문건수

반환값 : 정수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

 **GetBidByCode(sItemCode, nAskLevel)**

설명 : 종목코드(단축코드)로 지정한 옵션의 매수호가

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

nAskLevel – 정수, 1~5 : 매수1호가~매수5호가

 **GetBidAmountByCode(sItemCode, nAskLevel)**

설명 : 종목코드(단축코드)로 지정한 옵션의 매수호가 잔량

반환값 : 정수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

nAskLevel – 정수, 1~5 : 매수1호가~매수5호가

 **GetBidCountByCode(sItemCode, nAskLevel)**

설명 : 종목코드(단축코드)로 지정한 옵션의 매수호가 건수

반환값 : 정수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

nAskLevel – 정수, 1~5 : 매수1호가~매수5호가

 **GetBidTotalAmountByCode(sItemCode)**

설명 : 종목코드(단축코드)로 지정한 옵션의 매수호가 총잔량

반환값 : 정수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

 **GetBidTotalCountByCode(sItemCode)**

설명 : 종목코드(단축코드)로 지정한 옵션의 매수호가 총주문건수

반환값 : 정수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

 **GetCurrentByCode(sItemCode)**

설명 : 종목코드(단축코드)로 지정한 옵션의 현재가

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

 **GetHighByCode(sItemCode)**

설명 : 종목코드(단축코드)로 지정한 옵션 종목의 당일고가

반 환 값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetLowByCode(sItemCode)

설 명 : 종목코드(단축코드)로 지정한 옵션 종목의 당일저가

반 환 값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetOpenByCode(sItemCode)

설 명 : 종목코드(단축코드)로 지정한 옵션 종목의 시가

반 환 값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetVolumeByCode(sItemCode)

설 명 : 종목코드(단축코드)로 지정한 옵션 종목의 당일 거래량

반 환 값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetOpenInterestByCode(sItemCode)

설 명 : 종목코드(단축코드)로 지정한 옵션 종목의 당일 미결제약정

반 환 값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetExercisePriceByCode(sItemCode)

설 명 : 종목코드(단축코드)로 지정한 옵션 종목의 행사가

반 환 값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetExpectedPriceByCode(sItemCode)

설 명 : 종목코드(단축코드)로 지정한 옵션 종목의 예상체결가

반 환 값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetDeltaByCode(sItemCode)

설 명 : 종목코드(단축코드)로 지정한 옵션 종목의 델타

반 환 값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetGammaByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션 종목의 감마

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetThetaByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션 종목의 세타

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetVegaByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션 종목의 베가

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetDisparateRatioByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션 종목의 괴리를

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetImpliedVolatilityByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션 종목의 내재변동성

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetIntrinsicValueByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션 종목의 내재가치

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

▣ GetRemainDaysByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션 종목의 잔존일수

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

#### GetTimeValueByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션 종목의 시간가치

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

#### GetTheoryPriceByCode(sItemCode)

설명 : 종목코드(단축코드)로 지정한 옵션 종목의 이론가

반환값 : 실수

매개변수 : sItemCode – 문자열, 종목코드(단축코드)

### 10) 참조데이터객체

#### GetInvestorInfoByCategory(nCategory, nInvestor, nAmountQty, nBuySell, nIndex)

설명 : 시장수급데이터를 리턴

반환값 : 실수

매개변수 : nCategory – 정수, 시장구분(0 : KOSPI선물, 1: KOSPI옵션전체, 2:KOSPI, 3: KOSDAQ)

nInvestor	주체
1	외국인
2	개인
3	금융투자
4	투신
5	은행
6	보험
7	기타금융
8	연기금
9	사모펀드
10	국가지자체
11	기타
12	외인기타

nInvestor – 정수, 주체

nAmountQty – 정수, 수량/금액(0:수량, 1:금액)

nBuySell – 정수, 매도/매수(1:매도, 2: 매수)

nIndex – 정수, N일전, 0이상 사용가능, 정수로 입력

### 11) 그리드객체

#### cols

설명 : 그리드의 열갯수

반환값 : 정수

## rows

설명 : 그리드의 행갯수

반환값 : 정수

### AddRows(count)

설명 : 그리드에 행을 추가합니다.

반환값 : 없음

매개변수 : count – %의 매개변수 목록

### DeleteRows(row)

설명 : 그리드의 행을 삭제합니다.

반환값 : 없음

매개변수 : row – 삭제할 행의 인덱스를 입력합니다.

인덱스는 복수로 지정 가능합니다. DeleteRows(1,3,5)

### GetValue(col, row, count, horizon)

설명 : 지정한 셀의 값을 반환합니다.

반환값 : 없음

매개변수 : col – 정수, 열번호, 첫열은 0부터 시작합니다.

row – 정수, 행번호, 첫행은 0부터 시작합니다.

count – 정수

horizon – 불대수, 입력되는 값이 배열이나 객체처럼 다수의 값일 경우 행에 배열시킬지

열에 배열시킬지를 지정합니다. 기본값 True(행)

### InsertRows(row, count)

설명 : 그리드의 지정된 인덱스에 행을 삽입합니다.

반환값 : 없음

매개변수 : row – 정수, 행번호, 첫행은 0부터 시작합니다.

count – 정수, 추가할 행의 개수를 입력합니다. 기본값 1

### SetAlign(col, row, flag)

설명 : 지정한 셀의 정렬방식을 지정합니다.

반환값 : 없음

매개변수 : col – 정수, 열번호, 첫열은 0부터 시작합니다.

row – 정수, 행번호, 첫행은 0부터 시작합니다.

flag – 정수, 정렬속성, 상수값을 참조하시기 바랍니다.

### SetColorBack(col, row, color)

설명 : 지정한 셀의 배경색을 지정합니다.

반환값 : 없음

매개변수 : col – 정수, 열번호, 첫열은 0부터 시작합니다.

row – 정수, 행번호, 첫행은 0부터 시작합니다.

color – 정수, 색상

### SetColorText(col, row, color)

설명 : 지정한 셀의 글자색을 지정합니다.

반환값 : 없음

매개변수 : col – 정수, 열번호, 첫열은 0부터 시작합니다.

row – 정수, 행번호, 첫행은 0부터 시작합니다.

color – 정수, 색상

### SetColWidth(col, width)

설명 : 지정한 열의 너비를 지정합니다.

반환값 : 없음

매개변수 : col – 정수, 열번호, 첫열은 0부터 시작합니다.

width – 정수, 열 너비

### SetValue(col, row, value, horizon)

설명 : 지정한 셀 혹은 셀들에 값을 입력합니다.

반환값 : 없음

매개변수 : col – 정수, 열번호, 첫열은 0부터 시작합니다.

row – 정수, 행번호, 첫행은 0부터 시작합니다.

value – 없음. 셀에 입력할 값을 지정합니다. 문자열, 숫자, 배열요소가 객체나 배열이 아닌 배열,

배열이나 객체를 프로퍼티로 갖지 않는 객체들을 지정할 수 있습니다

horizon – 불대수, 입력되는 값이 배열이나 객체처럼 다수의 값일 경우 행에 배열시킬지

열에 배열시킬지를 지정합니다. 기본값 True(행)

## 11) 체결통보객체

### accountNum

설명 : 계좌번호

반환값 : 문자열

 accountKind

설명 : 계좌종류

반환값 : 정수, (0: 위탁, 1:저축, 2: KP선물옵션)

 code

설명 : 종목코드(단축코드)

반환값 : 문자열

 fillCount

설명 : 체결수량

반환값 : 정수

 fillPrice

설명 : 체결가격

반환값 : 실수

 error

설명 : 에러메세지

반환값 : 문자열

 loanDate

설명 : 대출일

반환값 : 정수

 loanKind

설명 : 대출상세분류(0:보통, 1:유통용자매수, 3: 자기용자매수)

반환값 : 정수

 orderCondition

설명 : 주문조건

반환값 : 정수, (0: 없음, 1: IOC, 2: FOK)

 orderCount

설명 : 주문수량

반환값 : 정수

 orderKind

설명 : 주문구분

반환값 : 정수(1 : 매도, 2 : 매수)

 orderNum

설명 : 주문번호

반환값 : 문자열

 orderPrice

설명 : 주문가격

반환값 : 실수

 orgOrderNum

설명 : 원주문번호

반환값 : 문자열

 priceKind

설명 : 가격구분

반환값 : 정수

nPriceKind	가격구분
0	지정가
1	시장가
2	최유리지정가
3	지정가(IOC)
4	시장가(IOC)
5	최유리지정가(ICO)
6	지정가(FOK)
7	시장가(FOK)
8	최유리지정가(FOK)
9	조건부지정가
10	최우선지정가
11	장개시전 시간외 종가
12	시간외 종가(장종료 후)
13	시간외 단일가 매매

12) 주문응답객체

 accountNum

설명 : 계좌번호

반환값 : 문자열

#### code

설명 : 종목코드(단축코드)

반환값 : 문자열

#### error

설명 : 에러메세지

반환값 : 문자열

#### isNormal

설명 : 정상여부(주문이 정상접수 되면 true, 예러면 false)

반환값 : 불대수

#### loanDate

설명 : 대출일

반환값 : 정수

#### loanKind

설명 : 대출상세분류

반환값 : 정수(0:보통, 1:유통융자매수, 3:자기융자매수)

#### orderCondition

설명 : 주문조건

반환값 : 정수, (0: 없음, 1: IOC, 2: FOK)

#### orderCount

설명 : 주문수량

반환값 : 정수

#### orderId

설명 : 주문식별번호, Buy나 Sell함수를 사용했을 때 반환되는 값

반환값 : 정수

#### orderKind

설명 : 주문구분

반환값 : 정수(1 : 매도, 2 : 매수)

 orderNum

설명 : 주문번호

반환값 : 문자열

 orderPrice

설명 : 주문가격

반환값 : 실수

 orgOrderNum

설명 : 원주문번호

반환값 : 문자열

 priceKind

설명 : 가격구분

반환값 : 정수

priceKind	KRX	NXT	SOR(스마트)
0 : 지정가	○	○	○
1 : 시장가	○		
2 : 최유리지정가	○	○	○
3 : 지정가(IOC)	○		
4 : 시장가(IOC)	○	○	○
5 : 최유리지정가(IOC)	○		
6 : 지정가(FOK)	○		
7 : 시장가(FOK)	○	○	○
8 : 최유리지정가(FOK)	○		
9 : 조건부지정가	○		
10 : 최우선지정가	○	○	○
11 : 시간외종가(장개시 전)	○		
12 : 시간외종가(장종료 후)	○	○	
13 : 시간외단일가 매매	○		
14 : 중간가	○	○	
15 : 스탑지정가	○	○	

13) 완성신호액체

 apply

설명 : 시스템 적용구분

반환값 : 정수, (0 : 시험적용, 1 : 경보 후 주문, 2 : 자동주문)

 code

설명 : 완성신호가 발생한 종목코드(단축코드)

반환값 : 문자열

 count - 완성신호 주문수량

설명 : 주문수량

반환값 : 정수

 date

설명 : 완성신호가 발생한 날짜

반환값 : 정수

 name

설명 : 완성 신호명

반환값 : 문자열

 price

설명 : 신호가격

반환값 : 실수

 signalKind

설명 : 완성신호 종류

반환값 : 정수, (1 : Buy, 2 : ExitLong, 3 : Sell, 4 : ExitShort)

 signalType

설명 : 완성주문신호 유형

반환값 : 정수, (1 : OnClose, 2 : AtMarket, 3 : AtLimit, 4 : AtStop)

 time

설명 : 완성신호가 발생한 시간

반환값 : 정수

#### 14) 미완성신호객체

##### code

설명 : 미완성 신호가 발생한 종목코드(단축코드)

반환값 : 문자열

##### count

설명 : 주문수량

(※ 미완성 신호의 수량은 시스템 트레이딩 설정창의 비용/수량탭에서 설정한 경우에는 -1이 리턴됩니다. 작성시 유의하시기 바랍니다.)

반환값 : 정수

##### date

설명 : 미완성신호가 발생한 날짜

반환값 : 정수

##### name

설명 : 신호명

반환값 : 문자열

##### price

설명 : 신호가격

반환값 : 실수

##### signalKind

설명 : 미완성신호 종류

반환값 : 정수, (1 : Buy, 2 : ExitLong, 3 : Sell, 4 : ExitShort)

##### signalType

설명 : 미완성주문신호 유형

반환값 : 정수, (1 : OnClose, 2: AtMarket, 3: AtLimit, 4: AtStop)

##### time

설명 : 신호시간

반환값 : 정수

#### 15) 미체결내역객체

 code

설명 : 종목코드(단축코드)

반환값 : 문자열

 condition

설명 : 주문조건

반환값 : 정수, (0: 없음, 1: IOC, 2: FOK)

 count

설명 : 미체결수량

반환값 : 정수

 orderKind

설명 : 주문구분

반환값 : 정수, (1 : 매도, 2 : 매수)

 orderNum

설명 : 주문번호

반환값 : 문자열

 orgOrderNum

설명 : 원주문번호

반환값 : 문자열

 Price

설명 : 주문가격

반환값 : 실수

 priceKind

설명 : 가격구분

반환값 : 정수

nPriceKind	가격구분
0	지정가
1	시장가
2	최유리지정가
3	지정가(IOC)
4	시장가(IOC)
5	최유리지정가(ICO)
6	지정가(FOK)
7	시장가(FOK)
8	최유리지정가(FOK)
9	조건부지정가
10	최우선지정가
11	장개시전 시간외 종가
12	시간외 종가(장종료 후)
13	시간외 단일가 매매

#### 16) 잔고액체(Balance)

assessedAmount

설명 : 잔고평가금액

반환값 : 실수

avgUnitCost

설명 : 잔고평균단가

반환값 : 실수

code

설명 : 잔고의 종목코드(단축코드)

반환값 : 문자열

count

설명 : 잔고수량

반환값 : 정수

current

설명 : 종목의 현재가

반환값 : 실수

position

설명 : 주문구문

반환값 : 정수, (1:매도, 2:매수)

## 17) 확장차트 종목설정 객체 (ReqChartItem)

수식 내에서 차트를 생성하기 위해 차트생성에 필요한 기본정보를 지정하는 객체입니다.

- new ReqChartItem(**code**, cycle, period, count, countKind, modifyPrice, dailyGap)  
종목코드만 지정 주식은 KRX, 선옵은 주간입니다.
- new ReqChartItem({**code** : “종목코드”, **marketKind**:시장구분:, **exchangeKind**:거래소},  
cycle, period, count, countKind, modifyPrice, dailyGap)

주식에서 거래소(통합, KRX,NXT)를 지정하거나 선물옵션에서 시장(주간,야간,복합)을 지정하려면 code에 JSON객체로 종목정보를 담아 지정해야 합니다.

### code

설명 : 종목코드 또는 종목정보를 지정하는 객체

- ◆ 종목코드만 지정  
주식은 KRX, 선옵은 주간입니다.
- ◆ 주식에서 거래소(통합, KRX,NXT)를 지정하거나 선물옵션에서 시장(주간,야간,복합)을 지정하려면 item에 JSON객체로 종목정보를 담아 지정해야 합니다.

{**code** : “종목코드” , **exchangeKind** : 상수 , **marketKind**: 상수}

\* JSON객체는 중괄호{ }안에 key:value로 이루어진 쌍들을 콤마(,) 나열해서 표현합니다.

	Key	value
종목코드	code	문자열
거래소	exchangeKind	상수 (1:통합, 2:KRX, 4:NXT)
시장구분	marketKind	상수 통합 - 1, CHART_MARKET_CMPLX 주간 - 2, CHART_MARKET_Day 야간 - 4, CHART_MARKET_NIGHT

자료형 : 문자열 또는 객체

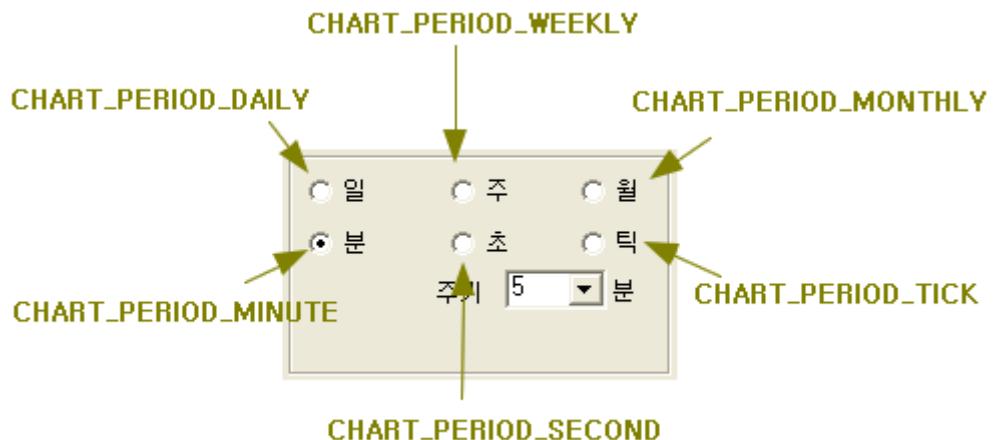
### cycle

설명 : 주기

틱(1~1000), 초(1~60), 분(1~400), 일(1), 주(1), 월(1)로 입력합니다.

자료형 : 정수

### Period



설명 : 차트주기, 기간구분

틱 - CHART\_PERIOD\_TICK

초 - CHART\_PERIOD\_SECOND

분 - CHART\_PERIOD\_MINUTE

일 - CHART\_PERIOD\_DAILY

주 - CHART\_PERIOD\_WEEKLY

월 - CHART\_PERIOD\_MONTHLY

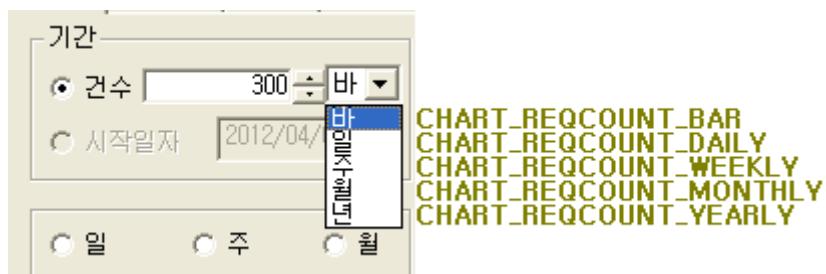
자료형 : 정수

#### count

설명 : 조회건수(1~5000)

자료형 : 정수

#### countKind



설명 : 조회건수 구분

바 - CHART\_REQCOUNT\_BAR

일 - CHART\_REQCOUNT\_DAILY

주 - CHART\_REQCOUNT\_WEEKLY

월 - CHART\_REQCOUNT\_MONTHLY

년 - CHART\_REQCOUNT\_YEARLY

자료형 : 정수

#### modifyPrice

설명 : 수정주가

적용함(true), 적용안함(false)

자료형 : 불대수

▣ dailyGap

설명 : 갭보정

적용함(true), 적용안함(false)

자료형 : 불대수

▣ ReqChartItem(code, cycle, Period, count, countKind, modifyPrice, dailyGap)

※코드만 지정

```
//기본종목셋팅(연결선을 주간장,5분 5000개, 갭보정안함, 수정주가처리 안함)
var ChartSet = new ReqChartItem("00000000",5,CHART_PERIOD_MINUTE,
    5000,CHART_REQCOUNT_BAR, false, false);

//시스템 셋팅

var SysSet = new SystemInfo("Stochastics K_D",YL_TYPE_NORMAL);

//지표 셋팅(지표는 복수로 지정이 가능하므로 Array에 담아 지정)
var IndSet = new Array(new IndicatorInfo("Stochastics"),new IndicatorInfo("ADX"));

//참조데이터 셋팅(ReqChartItem를 이용해 종목과 주기등지정)
//data2종목(삼성전자,5분봉 5000개,갭보정안함, 수정주가처리 안함)
var Data2 = new ReqChartItem("005930",5,CHART_PERIOD_MINUTE,
    5000,CHART_REQCOUNT_BAR,false,false);

//data3종목(SK하이닉스,5분봉 5000개,갭보정안함, 수정주가처리 안함)
var Data3 = new ReqChartItem("000660",5,CHART_PERIOD_MINUTE,
    5000,CHART_REQCOUNT_BAR,false,false);

//data3종목(종합주가지수,5분봉 5000개,갭보정안함, 수정주가처리 안함)
var Data4 = new ReqChartItem("001",5,CHART_PERIOD_
    MINUTE,5000,CHART_REQCOUNT_BAR,false,false);

//참조 데이터는 복수로 지정이 가능하므로 Array에 담아 지정
var RefSet = new Array(Data2,Data3,Data4);

//차트 셋팅 제외하고 시스템,지표, 참조데이터는 생략가능, 생략하면 null로 지정
Main.ReqChartEx(ChartSet, SysSet, IndSet, RefSet);
```

▣ ReqChartItem({code : “종목코드”, marketKind:시장구분:, exchangeKind:거래소},

cycle, period, count, countKind, modifyPrice, dailyGap)

※ JSON객체로 지정

```
//기본종목셋팅({연결선을 복합장 KRX},5분 5000개, 갭보정안함, 수정주가처리 안함)
```

```

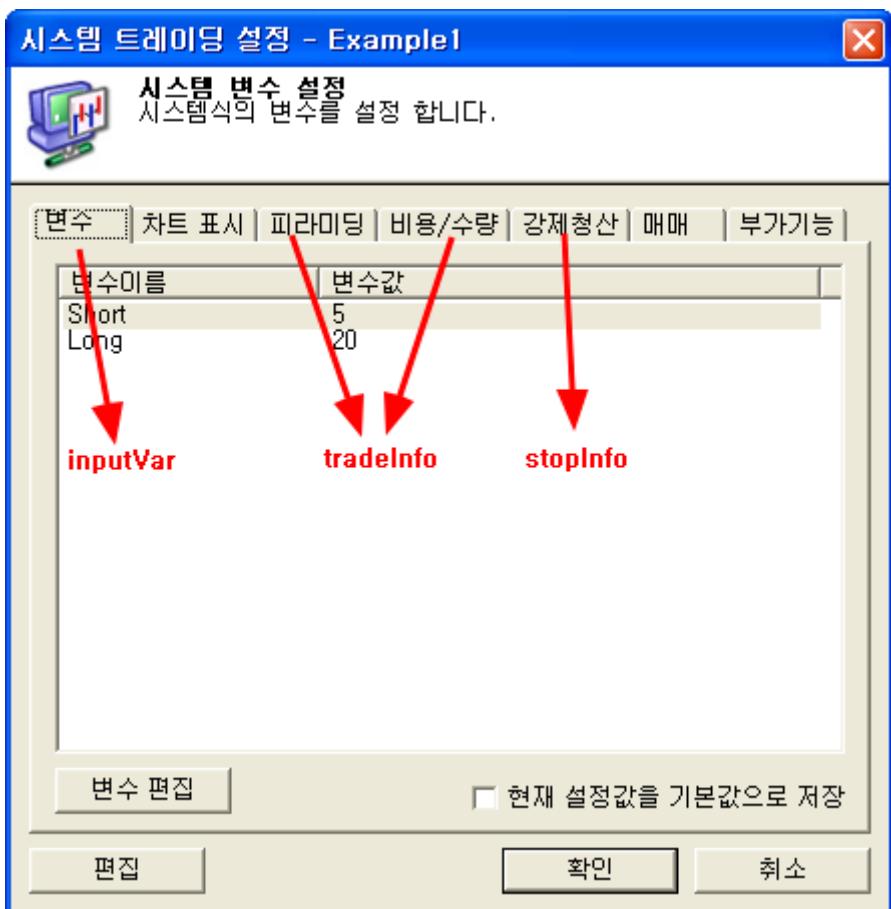
var item1 = {code:"00000000", marketKind:CHART_MARKET_CMPLX}
var ChartSet = new ReqChartItem(item1,5,CHART_PERIOD_MINUTE,
                                5000,CHART_REQCOUNT_BAR,false, false);
//시스템 셋팅
var SysSet = new SystemInfo("Stochastics K_D",YL_TYPE_NORMAL);
//지표 셋팅(지표는 복수로 지정이 가능하므로 Array에 담아 지정)
var IndSet = new Array(new IndicatorInfo("Stochastics"),new IndicatorInfo("ADX"));
//참조데이터 셋팅(ReqChartItem을 이용해 종목과 주기등지정) /
//data2종목({ 삼성전자 통합차트},5분봉 5000개,캡보정안함, 수정주가처리 안함)
var item2 = {code:"005930", exchangeKind:1}
var Data2 = new ReqChartItem(item2,5,CHART_PERIOD_MINUTE,
                             5000,CHART_REQCOUNT_BAR,false,false)
//data3종목({ SK하이닉스 통합차트},5분봉 5000개,캡보정안함, 수정주가처리 안함)
var item3 = {code:"000660", exchangeKind:1}
var Data3 = new ReqChartItem(item3,5,CHART_PERIOD_MINUTE,
                             5000,CHART_REQCOUNT_BAR,false,false)
//data4종목({ 종합주가지수 KRX차트},5분봉 5000개,캡보정안함, 수정주가처리 안함)
var item4 = {code:"001",exchangeKind:2}
var Data4 = new ReqChartItem(item4,5,CHART_PERIOD_MINUTE,
                             5000,CHART_REQCOUNT_BAR,false,false)
//참조 데이터는 복수로 지정이 가능하므로 Array에 담아 지정
var RefSet = new Array(Data2,Data3,Data4);
//차트 셋팅 제외하고 시스템,자표, 참조데이터는 생략가능, 생략하면 null로 지정
Main.ReqChartEx(ChartSet, SysSet, IndSet, RefSet);

```

#### 18) 확장차트 시스템설정 객체 (SystemInfo)

수식 내에서 생성된 차트에 시스템을 적용하기 위한 기본정보를 지정하는 객체입니다.

적용할 시스템명과 파일종류를 제외하고 시스템의 변수나 피라미딩, 비용/수량, 강제청산은 지정해야 할 것이 다수이므로 각각을 하나의 객체에 정보를 담아 지정해 주게 됩니다.



시스템은 시험적용으로 적용됩니다. 주문은 모두 스팟내에서 처리해야 합니다.

그러므로 따로 매매탭이나 부가기능을 설정하는 부분은 없습니다.

#### name

설명 : 시스템 이름.

자료형 : 문자열

#### Kind

설명 : 시스템 파일의 종류를 지정

일반수식(YL\_TYPE\_NORMAL)

배포버전수식(YL\_TYPE\_EXPORTED)

합성시스템(YL\_TYPE\_STRATEGY)

ez\_전략생성기(YL\_TYPE\_WIZARD)

자료형 : 정수

#### inputVar

설명 : 시스템식의 Input변수 정보.

변경을 원하는 변수의 이름과 값을 YLInputVar 객체로 지정합니다.( YLInputVar 객체 참조)

자료형 : Array

#### tradeInfo

설명 : 피라미딩 및 거래/비용 설정,  
시스템 적용에 필요한 피라미딩 정보와 비용/수량 관련 설정을  
SystemTradeInfo 객체를 이용해 지정합니다. (SystemTradeInfo 객체 참조)  
자료형 : 객체

#### stopInfo

설명 : 강제청산 설정.  
시스템 적용에 필요한 강제청산 정보를 SystemStopInfo 객체로 이용해 지정합니다.  
(SystemStopInfo 객체 참조)  
자료형 : 객체

#### SystemInfo(name,kind,inputVar,tradeInfo,stopInfo)

설명 : SystemInfo 객체 생성자  
반환값 : 객체  
매개변수 : name – 문자열, 시스템명  
kind – 정수, 시스템 파일 종류.  
일반수식(YL\_TYPE\_NORMAL) 배포버전수식(YL\_TYPE\_EXPORTED)  
합성시스템(YL\_TYPE\_STRATEGY) ex-전략생성기(YL\_TYPE\_WIZARD)  
inputVar – Array, 적용할 시스템의 변수 설정, YLInputVar 객체로 지정합니다. 기본값 : null  
( YLInputVar 객체 참조)  
tradeInfo – 객체, 적용할 시스템의 피라미딩, 비용/수량 설정,  
SystemTradeInfo 객체로 이용해 지정합니다. 기본값 : null, (SystemTradeInfo 객체 참조)  
StopInfo – 객체, 적용할 시스템의 강제청산 설정  
SystemStopInfo 객체로 이용해 지정합니다. 기본값 : null, (SystemStopInfo 객체 참조)

### 19) 확장차트 수식적용 Input변수 객체 (YLInputVar)

수식 내에서 생성한 차트에 시스템이나 지표를 적용할 때 외부변수를 설정하기 위한 객체입니다.

#### name

설명 : 변수명  
자료형 : 문자열

#### value

설명 : 변수값  
자료형 : 문자열

 YLInputVar(name,value)

설명 : YLInputVar 객체 생성자

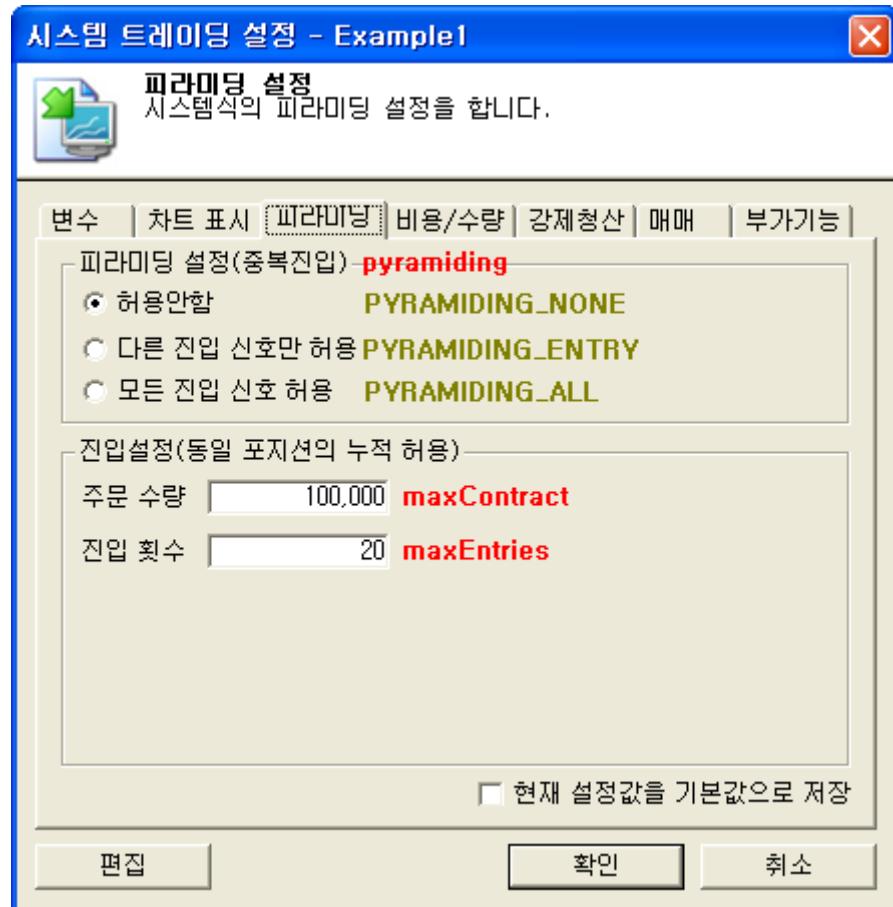
반환 값 : 객체

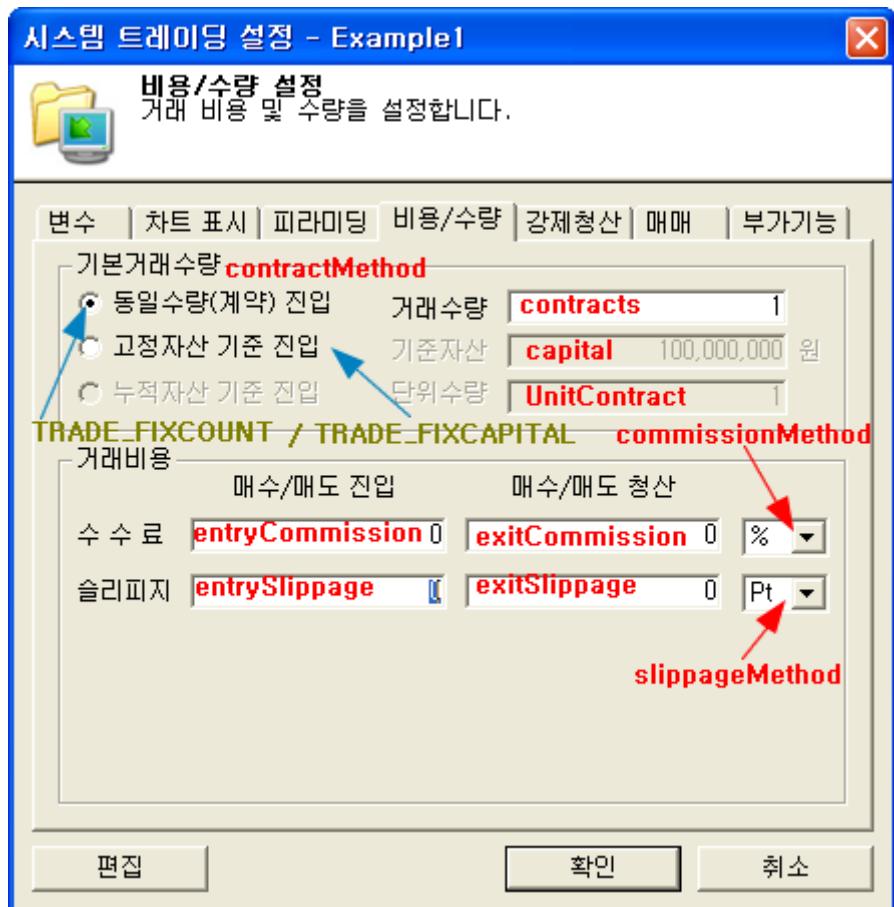
매개변수 : name – 문자열, 변수명

value – 문자열, 변수값

20) 확장차트 시스템적용 피라미딩, 비용/수량 설정 객체 (SystemTradeInfo)

수식 내에서 생성한 차트에 시스템 적용 시 피라미딩과 비용/수량을 설정하기 위한 객체입니다.





#### contractMode

설명 : 기본거래수량 설정

동일수량(계약) 진입 : TRADE\_FIXCOUNT(기본값)

고정자산기준 진입 : TRADE\_FIXCAPITAL

자료형 : 정수

#### contract

설명 : 기본거래수량 - 동일수량(계약) 진입 시 거래수량

자료형 : 정수

#### capital

설명 : 기본거래수량 - 고정자산 기준 진입 시 기준자산

자료형 : 실수

#### unitContract

설명 : 기본거래수량 - 고정자산 기준 진입 시 단위수량

자료형 : 정수

 entryCommission

설명 : 거래비용 - 진입수수료

자료형 : 실수

 exitCommission

설명 : 거래비용 - 청산수수료

자료형 : 실수

 commissionMethod

설명 : 거래비용 - 수수료 계산단위

% : CALCMETHOD\_PERCENT

Pt : CALCMETHOD\_POINT

틱 : CALCMETHOD\_TICK

자료형 : 정수

 entrySlippage

설명 : 거래비용 - 진입슬리피지

자료형 : 실수

 exitSlippage

설명 : 거래비용 - 청산슬리피지

자료형 : 실수

 slippageMethod

설명 : 거래비용 슬리피지 단위

% : CALCMETHOD\_PERCENT

Pt : CALCMETHOD\_POINT

틱 : CALCMETHOD\_TICK

자료형 : 정수

 pyramiding

설명 : 피라미딩 설정

허용안함 : PYRAMIDING\_NONE

모든진입신호허용 : PYRAMIDING\_ALL

다른진입신호만허용 : PYRAMIDING\_ENTRY

자료형 : 정수

▣ maxContract

설명 : 진입설정(동일포지션누적허용) 주문수량

자료형 : 정수

▣ maxEntries

설명 : 진입설정(동일포지션누적허용) 진입횟수

자료형 : 정수

▣ SystemTradeInfo(ContractMode, Contracts, capital, UnitContract, entryCommission, exitCommission, commissionMethod, entrySlippage, exitSlippage, slippageMethod, pyramiding, maxContract, maxEntries)

설명 : SystemTradeInfo 객체 생성자

반환값 : 객체

매개변수 : contractMode – 정수, 기본거래수량 설정(기본값 : TRADE\_FIXCOUNT, 동일수량(계약) 진입)

contracts – 정수, 기본거래수량(동일수량(계약) 진입 시 거래수량, 기본값 : 1)

capital – 실수, 기본거래수량(고정자산 기준 진입시 기준자산, 기본값 : 100,000,000)

UnitContract – 정수, 기본거래수량(고정자산 기준 진입시 단위수량, 기본값: 1)

entryCommission – 실수, 진입수수료(기본값 : 0)

exitCommission – 실수, 청산수수료(기본값 : 0)

commissionMethod – 정수, 수수료 계산단위(기본값 : CALCMETHOD\_PERCENT)

entrySlippage – 실수, 진입슬리피지(기본값 : 0)

exitSlippage – 실수, 청산슬리피지(기본값 : 0)

slippageMethod – 정수, 슬리피지 계산단위(기본값 : CALCMETHOD\_POINT)

pyramiding – 정수, 피라미딩설정,(기본값 : PYRAMIDING\_NONE)

maxContract – 정수, 동일포지션 누적허용 주문수량 (기본값 : 100,000)

maxEntries – 정수, 동일포지션 누적허용 진입횟수(기본값 : 20)

## 21) 확장차트 시스템적용 강제청산 설정 객체 (SystemStopInfo)

수식 내에서 생성한 차트에 시스템 적용할 때 강제청산을 설정하기 위한 객체입니다.

강제청산별로 설정해야 할 내용이 다수이므로 청산별로 설정할 수 있는 객체가 제공됩니다.

해당 객체를 이용해 설정합니다.



#### ■ StopLoss

설명 : 손절매 설정

자료형 : 객체

#### ■ StopTrailing

설명 : 최고수익대비하락 설정

자료형 : 객체

#### ■ StopTargetProfit

설명 : 목표수익 설정

자료형 : 객체

#### ■ StopInactivity

설명 : 최소가격변수 설정

자료형 : 객체

#### StopEndDay

설명 : 당일청산 설정

자료형 : 객체

#### StopTiming

설명 : 청산시점 설정 (기본값 : STOP\_INSTANT STOP\_BARFINISH)

자료형 : 객체

#### SystemStopInfo(StopLoss, StopTrailing, StopTargetProfit, StopInactivity, StopEndDay, StopTiming)

설명 : SystemTradeInfo 객체 생성자

반환값 : 객체

매개변수 : StopLoss – 손절매. 기본값 : null

StopTrailing – 객체, 최대수익대비하락(기본값 : null)

StopTargetProfit – 객체, 목표수익(기본값 : null)

StopInactivity – 객체, 최소가격변화(기본값 : null)

StopEndDay – 객체, 당일청산(기본값 : null)

StopTiming – 정수, 청산시점(기본값 : STOP\_INSTANT)

## 22) 확장차트 시스템적용 손절매 설정 객체 (StopLoss)

수식 내에서 생성한 차트에 시스템 적용할 때 강제청산 중 손절매를 설정하기 위한 객체입니다.

#### downPrice

설명 : 손절매

자료형 : 실수

#### calcMethod

설명 : 손절매 계산단위

% : CALCMETHOD\_PERCENT

Pt : CALCMETHOD\_POINT

자료형 : 정수

#### color

설명 : 손절매 표시색

자료형 : 정수

#### StopLoss(downPrice, calcMethod, color)

설명 : StopLoss 객체 생성자

반 환 값 : 객체

매개변수 : downPrice – 실수, 손절매(필수입력)

calcMethod – 정수, 손절매 계산단위(기본값 : CALCMETHOD\_PERCENT)

color – 정수, (손절매 표시색, 기본색 : 빨강)

### 23) 확장차트 시스템적용 목표수익 설정 객체 (StopProfitTarget)

수식 내에서 생성한 차트에 시스템 적용할 때 강제청산 중 목표수익을 설정하기 위한 객체입니다.

 TargetProfit

설명 : 목표수익

자료형 : 실수

 calcMethod

설명 : 목표수익 계산단위

% : CALCMETHOD\_PERCENT

Pt : CALCMETHOD\_POINT

자료형 : 정수

 color

설명 : 목표수익 표시색

자료형 : 정수

 StopProfitTarget(TargetProfit, calcMethod, color)

설명 : TargetProfit 객체 생성자

반환값 : 객체

매개변수 : TargetProfit – 실수, 목표수익(필수입력)

calcMethod – 정수, 목표수익 계산단위(기본값 : CALCMETHOD\_PERCENT)

color – 정수, 목표수익 표시색(기본색 : 빨강)

### 24) 확장차트 시스템적용 최대수익대비하락 설정 객체 (StopTrailing)

수식 내에서 생성한 차트에 시스템 적용할 때 강제청산 중 최고수익대비하락을 설정하기 위한 객체입니다.

 downPrice

설명 : 수익감소값

자료형 : 실수

 minProfot

설명 : 최소수익값

자료형 : 실수

▣ calcMethod

설명 : 최대수익대비하락 계산단위

% : CALCMETHOD\_PERCENT

Pt : CALCMETHOD\_POINT

자료형 : 정수

▣ maxProfitMode

설명 : 최대수익기준. 종류 : 수익대비(0), 최고가격대비(1)

자료형 : 정수

▣ color

설명 : 최대수익대비하락 표시색

자료형 : 정수

▣ StopTrailing(downPrice, minProfot ,calcMethod, maxProfitMode, color)

설명 : StopTrailing 객체 생성자

반환값 : 객체

매개변수 : downPrice – 실수, 수익감소값(필수입력)

minProfot – 실수, 최소수익값(필수입력)

calcMethod – 정수, 최대수익대비하락 계산단위(기본값 : CALCMETHOD\_PERCENT)

maxProfitMode – 정수, 최대수익기준(기본값 : 수익대비)

color – 정수, 손절매 표시색(기본색 : 빨강)

## 25) 확장차트 시스템적용 최소가격변화 설정 객체 (StopInactivity)

수식 내에서 생성한 차트에 시스템 적용할 때 강제청산 중 최소가격변화를 설정하기 위한 객체입니다.

▣ changePrice

설명 : 최소변화가격

자료형 : 실수

▣ inBar

설명 : 최소변화 봉수, 1~150봉

자료형 : 정수

▣ calcMethod

설명 : 최소가격변화 계산단위

% : CALCMETHOD\_PERCENT

Pt : CALCMETHOD\_POINT

자료형 : 정수

#### ■ color

설명 : 최소가격변화 표시색

자료형 : 정수

#### ■ StopInactivity(changePrice, inBar, calcMethod, color)

설명 : StopInactivity 객체 생성자

반환값 : 객체

매개변수 : changePrice – 실수, 최소변화가격(필수입력)

inBar – 정수, 최소변화 봉수(필수입력)

calcMethod – 정수, 최소가격변화 계산단위(기본값 : CALCMETHOD\_PERCENT)

color – 정수, 최소가격변화 표시색(기본색 : 빨강)

### 26) 확장차트 시스템적용 당일청산 설정 객체 (StopEndOfDay)

수식 내에서 생성한 차트에 시스템 적용할 때 강제청산 중 당일청산을 설정하기 위한 객체입니다.

#### ■ time

설명 : 당일청산시간, HHMMDD

자료형 : 정수

#### ■ color

설명 : 당일청산 표시색

자료형 : 정수

#### ■ StopEndOfDay(time, color)

설명 : StopEndOfDay 객체 생성자

반환값 : 객체

매개변수 : time – 정수, 당일청산시간(필수입력)

color – 정수, 당일청산 표시색(기본색 : 빨강)

### 27) 확장차트 지표 설정 객체 (IndicatorInfo)

수식 내에서 생성한 차트에 지표를 설정하기 위한 객체입니다.

#### ■ name

설명 : 적용할 지표명을 지정합니다.

자료형 : 문자열

#### ▣ Kind

설명 : 지표파일의 종류를 지정합니다.

일반파일 : YL\_TYPE\_NORMAL

배포버전 : YL\_TYPE\_EXPORTED

자료형 : 정수

#### ▣ inputVar

설명 : 지표식의 외부변수 정보. 변경을 원하는 변수의 이름과 값을 YLInputVar 객체로 지정합니다.

기본값 : null

자료형 : Array

#### ▣ IndicatorInfo(name, Kind, color)

설명 : IndicatorInfo 객체 생성자

반환값 : 객체

매개변수 : name – 문자열, 지표명

Kind – 정수, 지표파일종류(기본값 : YL\_TYPE\_NORMAL)

inputVar – Array, 지표식의 외부변수 정보,

변경을 원하는 변수의 이름과 값을 YLInputVar 객체로 지정합니다.

### 28) YesSpot Constants(상수)

스크립트 작성 편의를 위해 예스스팟에서 정의한 상수들입니다.

모두 대문자로 사용하셔야 합니다.

#### ▣ ONCLOSE

설명 : 차트신호유형 중 ONCLOSE를 나타내는 상수입니다.

완성신호객체(Signal)과 미완성신호객체(IncompleteSignal)의 signalType에 할당됩니다.

signalType의 유형은 (1 : OnClose, 2: AtMarket, 3: AtLimit, 4: AtStop)로

스크립트에서 if (signalType == 1)과 같이 작성해서 발생한 신호의 타입이 ONCLOSE임을 표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.

```
if (signalType == ONCLOSE)
```

#### ▣ ATMARKET

설명 : 차트신호유형 중 ATMARKET을 나타내는 상수입니다.

완성신호객체(Signal)에 할당됩니다.

signalType의 유형은 (1 : OnClose, 2: AtMarket, 3: AtLimit, 4: AtStop)로  
스크립트에서 if (signalType == 2)와 같이 작성해서 발생한 신호의 타입이 ATMARKET임을  
표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.

```
if (signalType == ATMARKET)
```

#### ■ ATLIMIT

설명 : 차트신호유형 중 ATLIMIT를 나타내는 상수입니다.

완성신호객체(Signal)에 할당됩니다.

signalType의 유형은 (1 : OnClose, 2: AtMarket, 3: AtLimit, 4: AtStop)로  
스크립트에서 if (signalType == 3)와 같이 작성해서 발생한 신호의 타입이 ATLIMIT 임을  
표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.

```
if (signalType == ATLIMIT)
```

#### ■ ATSTOP

설명 : 차트신호유형 중 ATSTOP을 나타내는 상수입니다.

완성신호객체(Signal)에 할당됩니다.

signalType의 유형은 (1 : OnClose, 2: AtMarket, 3: AtLimit, 4: AtStop)로  
스크립트에서 if (signalType == 4)와 같이 작성해서 발생한 신호의 타입이 ATSTOP임을  
표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.

```
if (signalType == ATSTOP)
```

#### ■ SIG\_BUY

설명 : 차트신호종류 중 BUY를 나타내는 상수입니다.

완성신호객체(Signal)과 미완성신호객체(IncompleteSignal)의 signalKind에 할당됩니다.

signalKind의 유형은 (1 : Buy, 2: Exitlong, 3: Sell, 4: ExitShort)으로  
스크립트에서 if (signalKind== 1)와 같이 작성해서 발생한 신호의 종류가 Buy임을  
표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.

```
if (signalKind == SIG_BUY)
```

#### ■ SIG\_EXITLONG

설명 : 차트신호종류 중 EXITLONG을 나타내는 상수입니다.

완성신호객체(Signal)과 미완성신호객체(IncompleteSignal)의 signalKind에 할당됩니다.

signalKind의 유형은 (1 : Buy, 2: Exitlong, 3: Sell, 4: ExitShort)으로  
스크립트에서 if (signalKind== 2)와 같이 작성해서 발생한 신호의 종류가 EXITLONG임을  
표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.

```
if (signalKind == SIG_EXITLONG)
```

### SIG\_SELL

설 명 : 차트신호종류 중 SELL을 나타내는 상수입니다.

완성신호액체(Signal)과 미완성신호액체(IncompleteSignal)의 signalKind에 할당됩니다.

signalKind의 유형은 (1 : Buy, 2: Exitlong, 3: Sell, 4: ExitShort)으로

스크립트에서 if (signalKind== 3)와 같이 작성해서 발생한 신호의 종류가 SELL임을 표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.

```
if (signalKind == SIG_SELL)
```

### SIG\_EXITSHORT

설 명 : 차트신호종류 중 EXITSHORT을 나타내는 상수입니다.

완성신호액체(Signal)과 미완성신호액체(IncompleteSignal)의 signalKind에 할당됩니다.

signalKind의 유형은 (1 : Buy, 2: Exitlong, 3: Sell, 4: ExitShort)으로

스크립트에서 if (signalKind== 4)와 같이 작성해서 발생한 신호의 종류가 EXITSHORT임을 표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.

```
if (signalKind == SIG_EXITSHORT)
```

### UID\_QUOTATION

설 명 : 종목의 업데이트 종류 중 시세 자동업데이트를 나타내는 상수입니다.

Main액체의 OnUpdateMarket(sItemCode, IUpdateID)이벤트의 IUpdateID에 할당됩니다.

IUpdateID의 종류는 (20001 : 시세 자동업데이트, 20002: 호가 자동업데이트)로

스크립트에서 if (IUpdateID== 20001)과 같이 작성해서 현재 발생한 업데이트가

시세 자동업데이트임을 표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.

```
if (IUpdateID == UID_QUOTATION)
```

### UID\_ASKSBIDS

설 명 : 종목의 업데이트 종류 중 호가 자동업데이트를 나타내는 상수입니다.

Main액체의 OnUpdateMarket(sItemCode, IUpdateID)이벤트의 IUpdateID에 할당됩니다.

IUpdateID의 종류는 (20001 : 시세 자동업데이트, 20002: 호가 자동업데이트)로

스크립트에서 if (IUpdateID== 20002)와 같이 작성해서 현재 발생한 업데이트가

호가 자동업데이트임을 표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.

```
if (IUpdateID == UID_ASKSBIDS)
```

### UID\_EXPECTPRICE

설 명 : (현재 준비중입니다)

종목의 업데이트 종류 중 예상체결 데이터의 업데이트를 나타내는 상수입니다.

Main 객체의 OnUpdateMarket(sItemCode, lUpdateID)이벤트의 lUpdateID에 할당됩니다.  
lUpdateID의 종류는 (20001 : 시세 자동업데이트, 20002: 호가 자동업데이트, 20003:예상체결데이터  
업데이트)로 스크립트에서 if (lUpdateID== 20003)와 같이 작성해서 현재 발생한 업데이트가  
예상체결데이터의 업데이트임을 표현하지만 상수를 이용해 다음과 같이 작성할 수도 있습니다.  
if (lUpdateID == UID\_EXPECTPRICE)

 CHART\_PERIOD\_TICK

설 명 : 차트기간구분의 틱을 의미합니다.

 CHART\_PERIOD\_SECOND

설 명 : 차트기간구분의 초를 의미합니다.

 CHART\_PERIOD\_MINUTE

설 명 : 차트기간구분의 분을 의미합니다.

 CHART\_PERIOD\_DAILY

설 명 : 차트기간구분의 일을 의미합니다.

 CHART\_PERIOD\_WEEKLY

설 명 : 차트기간구분의 주를 의미합니다.

 CHART\_PERIOD\_MONTHLY

설 명 : 차트기간구분의 월을 의미합니다.

 CHART\_REQCOUNT\_BAR

설 명 : 차트 조회건수 구분의 바를 의미합니다.

 CHART\_REQCOUNT\_DAILY

설 명 : 차트 조회건수 구분의 일을 의미합니다.

 CHART\_REQCOUNT\_WEEKLY

설 명 : 차트 조회건수 구분의 주를 의미합니다.

 CHART\_REQCOUNT\_MONTHLY

설 명 : 차트 조회건수 구분의 월을 의미합니다.

**■ CHART\_REQCOUNT\_YEARLY**

설명 : 차트 조회건수 구분의 년을 의미합니다.

**■ CALCMETHOD\_PERCENT**

설명 : 강제청산이나 수수료등의 수치를 계산할 때 %기준으로 설정합니다.

**■ CALCMETHOD\_POINT**

설명 : 강제청산이나 수수료등의 수치를 계산할 때 포인트나 원 기준으로 설정합니다.

**■ CALCMETHOD\_TICK**

설명 : 강제청산이나 수수료등의 수치를 계산할 때 틱 기준으로 설정합니다.

**■ PYRAMIDING\_ALL**

설명 : 피라미딩을 모든진입신호허용으로 설정

**■ PYRAMIDING\_ENTRY**

설명 : 피라미딩을 다른진입신호만허용으로 설정

**■ PYRAMIDING\_NONE**

설명 : 피라미딩을 허용안함으로 설정

**■ STOP\_INSTANT**

설명 : 강제청산의 청산시점을 조건만족즉시로 설정

**■ STOP\_BARFINISH**

설명 : 강제청산의 청산시점을 봉완선시로 설정

**■ TRADE\_FIXCOUNT**

설명 : 동일수량(계약) 진입

**■ TRADE\_FIXCAPITAL**

설명 : 고정자산기준 진입

**■ TRADE\_SUMCAPITAL**

설명 : 누적자산기준 진입

 YL\_TYPE\_NORMAL

설명 : 예스랭귀지 일반수식 파일

 YL\_TYPE\_EXPORTED

설명 : 예스랭귀지 배포버전 파일

 YL\_TYPE\_STRATEGY

설명 : 합성관리자로 만든 시스템 파일

 YL\_TYPE\_WIZARD

설명 : ez-전략생성기로 만든 시스템 파일

 XL\_BOTTOM

설명 : 텍스트를 셀의 아래쪽에 위치시킵니다

 XL\_CENTER

설명 : 텍스트를 셀의 중간에 위치시킵니다

 XL\_DISTRIBUTED

설명 : 텍스트를 셀에서 균등분할합니다.

 XL\_DOWNWARD

설명 : 텍스트를 아래로 회전합니다.

 XL\_HORIZONTAL

설명 : 텍스트를 수평으로 회전합니다.

 XL\_JUSTIFY

설명 : 텍스트를 셀에서 양쪽맞춤합니다.

 XL\_LEFT

설명 : 텍스트를 셀에서 왼쪽에 위치시킵니다.

 XL\_RIGHT

설명 : 텍스트를 셀에서 오른쪽에 위치시킵니다.

#### XL\_TOP

설명 : 텍스트를 셀의 위쪽에 위치시킵니다.

#### XL\_UPWARD

설명 : 텍스트를 위로 회전시킵니다.

#### XL\_VERTICAL

설명 : 텍스트를 세로쓰기합니다.

### 29) 내장 함수 및 객체

자바스크립트 자체에서 기본으로 제공하는 함수나 객체입니다.

전략 작성시에 많이 사용될 수 있는 몇몇 함수나 객체에 대해서만 간략히 설명을 드립니다.

좀더 자세한 내용은 자바스크립트관련 서적으로 참고하시기 바랍니다.

#### ● isNaN()

설명 : 입력값이 숫자가 아닌지를 판단하는 함수입니다.

숫자가 아니면 true가 리턴되고 숫자이면 false가 리턴됩니다.

isNaN("100") → false

isNaN("100k") → true

#### ● isFinite()

설명 : 숫자가 유한의 수이면 true 아니면 false를 반환합니다.

isFinite (12345) → true

isFinite ("12345") → true

isFinite (12.345) → true

isFinite ("YesSpot") → false

isFinite (12345우리나라) → false

isFinite ("5+6\*7") → false

#### ● Eval()

설명 : 문자열을 인자로 받아 이를 수식으로 변환하여 계산을 수행하고 결과를 리턴

str = "1+2+3+4"

result = eval(str); → result에 10 저장됨

#### ● parseFloat()

설명 : 문자열을 부동소수점 변환

```
result = parseFloat ("3.14"); → result에 3.14 저장됨
```

#### ● parseInt()

설명 : 문자열을 정수로 변환

```
result = parseFloat ("3.14"); → result에 3 저장됨
```

#### ● String()

설명 : 객체를 문자열로 변환하여 반환

```
result = String(3.14); → result에 3.14가 문자열로 저장됨
```

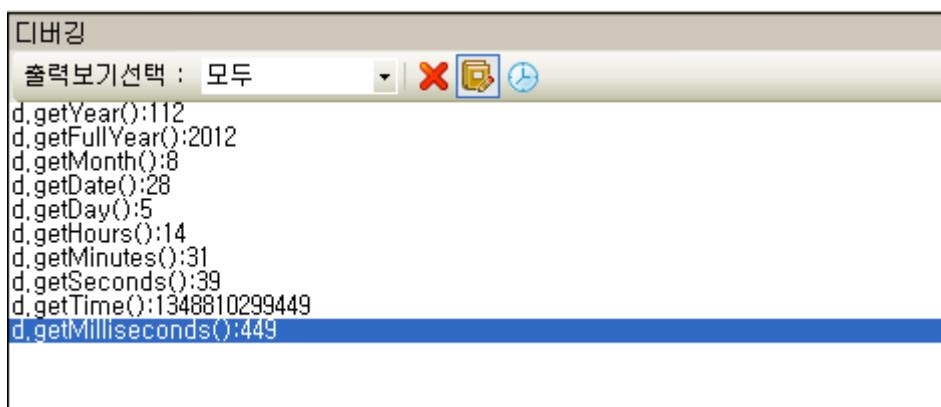
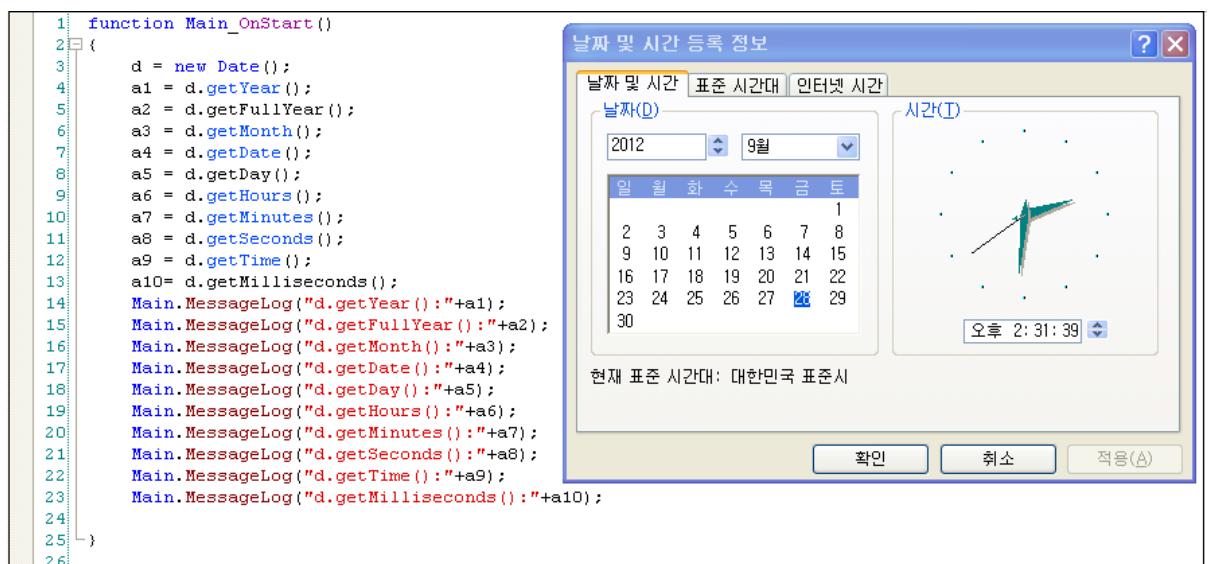
#### ● Number()

설명 : 값을 숫자로 변환하여 반환

```
result = Number("3.14"); → result에 3.14가 숫자로 저장됨
```

#### ● Date 객체의 주요 함수

getYear()	년도를 반환(1997 → 97, 2012 → 112)
getFullYear();	년도를 반환(YYYY)
getMonth();	월을 반환(1월을 0으로 시작, 12월이 11로 반환됨)
getDate();	일을 반환
getDay();	요일을 반환
getHours();	시간을 반환
getMinutes();	분을 반환
getSeconds();	초를 반환
getTime();	초를 반환(1970년1월1일 이후에 경과된 시간을 1000분의 1초로 반환)
getMilliseconds();	1000분의 1초로 표시



### ● Math 객체의 주요 함수

<code>abs(x)</code>	$x$ 의 절대값	<code>min(x,y)</code>	$x,y$ 중 작은 수
<code>ceil()</code>	올림	<code>pow(x,y)</code>	지수함수 $xy$
<code>cos(x)</code>	$x$ 의 cosine	<code>random()</code>	난수
<code>exp(x)</code>	$x$ 의 지수함수	<code>round(x)</code>	$x$ 를 반올림 한 수
<code>floor(x)</code>	내림	<code>sin(x)</code>	$x$ 의 sine
<code>Log(x)</code>	$x$ 의 로그함수	<code>sqrt(x)</code>	$x$ 의 제곱근
<code>max(x,y)</code>	$x,y$ 중 큰 수	<code>tan(x)</code>	$x$ 의 tangent

`Math.abs(-1)` → 1  
`Math.ceil(15.67)` → 16  
`Math.cos(90);` → -0.3380736161291701  
`Math.exp(16);` → 8886110.520507872

```

Math.floor(15.67);      → 15
Math.cos(90);           → 0.4480739161291701
Math.max(90,87);       → 90
Math.min(90,87);       → 87
Math.pow(2,3);          → 8
Math.random();          → 0.6762852098230685
Math.round(2.6);        → 3
Math.sin(90);           → 0.8939966636005578
Math.sqrt(4);            → 2
Math.tan(90);           → -1.995200412208242

```

### ● String 객체의 주요 함수

toLowerCase()	문자열을 소문자로 변환
toUpperCase()	문자열을 대문자로 변환
Substring(index1,index2)	Index1과 index2 사이에 있는 문자열을 리턴
concat(string)	두개의 문자열을 합쳐서 하나의 새로운 문자열로 만듬
IndexOf(string)	왼쪽부터 검색하여 첫번째 만나는 string의 위치를 리턴
charAt(index)	지정한 위치의 문자 리턴
Substr(start_index,Length)	문자열을 length만큼 잘라서 리턴

```

str1 = "YesStock";
str2 = "YesSpot";
str1.toLowerCase();      → yesstock
str1.toUpperCase();      → YESSTOCK
str1.charAt(2);          → s
str1.indexOf("o");       → 5
str1.substring(3,5);     → St
str1.substr(0, 5);       → YesSt
str1.concat(str2);       → YesStockYesSpot

```

## 5. 예스스팟 사용시 주의사항

1) 예스스팟은 스팟성의 데이터를 이용하여 전략을 만들고 자동매매가 가능한 기능입니다. 일반적인 시스템 트레이딩에서는 주문오류를 방지하기 위한 여러 가지 기능(예를 들어 진입신호가 발생되면 그 다음은 해당 진입신호에 대응되는 청산신호 발생)이 포함되어 있지만, 예스스팟에서는 사용자가 모든 조건을 직접 만들어서 사용해야 합니다. 따라서

로직을 잘못 작성할 경우 원하지 않는 주문이 반복해서 발생할 가능성이 있으므로 '시험적용' 상태로 충분히 검증한 이후에 사용하셔야 하며, 이후 발생되는 일체의 손해에 대해서 당사에 그 책임을 요구하지 않겠다는 것을 준수하셔야 합니다.

2) 예스스팟에서는 방대한 데이터를 제공하기 때문에 개별 데이터가 오류가 있을 수 있습니다. 따라서 개별적으로 사용하는 데이터의 원활한 수신여부와 정상적인 데이터 인지를 직접 검증해서 오류 여부를 판단하고 사용하셔야 됩니다. 예스스팟에서 제공하는 데이터에 관해서 회사는 데이터의 정확성을 보장하지 않으며 이 데이터를 이용하여 발생하는 일체의 손해에 대해서 당사에 그 책임을 요구하지 않겠다는 것을 준수하셔야 합니다.

3) 잔고(Balance), 미체결(Unfill), 체결(NotifyFill)등 계좌관련 객체 사용시 주의 사항

계좌참조 기능은 사용자 PC의 “가원장” 기능을 이용하는 것으로서 주문 접수/체결 응답이 실제 원장과 가원장에 도달하고 처리되는 시차와 데이터의 유실 등으로 인해 차이가 발생할 수 있습니다.

사용자는 반드시 적절한 시점에 잔고조회 등을 통해 가원장의 상태를 원장의 상태와 동기화 될 수 있도록 조치하여야 합니다. 기타 다른 매체나 단말을 통한 주문 및 입 출금 등 다른 요인도 존재합니다.

이런 현상에 대한 손실은 전적으로 사용자에게 귀속되며 회사는 이에 대한 보상이나 책임이 없습니다.

4) 시스템 자동매매는 서버와 접속이 유지된 상태에서만 정상적으로 처리되며 네트워크 장애나

사용자 컴퓨터상의 문제 등으로 인하여 접속이 끊어질 경우 동작하지 않습니다.

5) 시스템 전략의 조건을 만족하여 주문을 실행할 때 실제 매매체결로 꼭 이어지는 것은 아닙니다.

증거금 및 잔고 부족 등으로 주문이 거부 되거나 정상적인 주문의 경우에도 시장상황 및 매매체결의 원칙 등으로 체결이 보장되지 않습니다. 그러므로 주문/체결 내역조회를 반드시 확인하여야 합니다.

6) 주문응답이벤트(OnOrderResponse)는 스팟전략에서 발생한 주문에 대해서만 호출됩니다.

7) 예스스팟 스튜디오의 도구모음에서 를 클릭하면 스크립트 검증을 하게 됩니다.

해당 기능은 단순히 문법에 대해서만 검증합니다. 따라서, 쉼표나 괄호 등의 누락에 대한 부분은 검증할 수 있지만 선언되지 않은 변수의 사용 등의 실행에 대한 부분은 검증되지 않습니다.

8) 예스스팟은 서버에 과도한 부하를 줄이기 위해 시세조회와 계좌 조회를 15초당 60회로 제한하고 있습니다.

60회이상을 단기간에 요청하실 경우 시간타임을 주시고 요청을 해주셔야 합니다.

시간제한이 걸렸을 때는 Main.GetLimitedTime함수가 남은 시간(초)를 확인할 수 있습니다.

9) 예스스팟은 주문 오작동을 막기 위해 15초당 90회까지만 주문이 가능합니다.

90회이상을 단기간에 요청하실 경우 시간타임을 주시고 요청을 해주셔야 합니다.

시간제한이 걸렸을 때는 Main.GetLimitedTime함수가 남은 시간(초)를 확인할 수 있습니다.

## 6. 작성예제

스팟전략을 작성할 때는 전략 구현에 필요한 객체를 스크립트 객체화면에 추가해서 사용준비를 하고 스크립트 편집창에서 전략을 구현해 나가는 구조입니다.

※ 제공되는 모든 예제는 전략 작성시 참고용으로만 사용하시기 바랍니다.

### 예제1. 시스템 연동 타종목주문

예제1에서는 객체설정과 스팟전략 작성흐름과 작성법을 설명하기 위해 간단한 내용을 구현해 보도록 하겠습니다.

#### 1) 전략내용

선물차트에서 매수신호 발생시 ATM 등가격콜옵션 매수

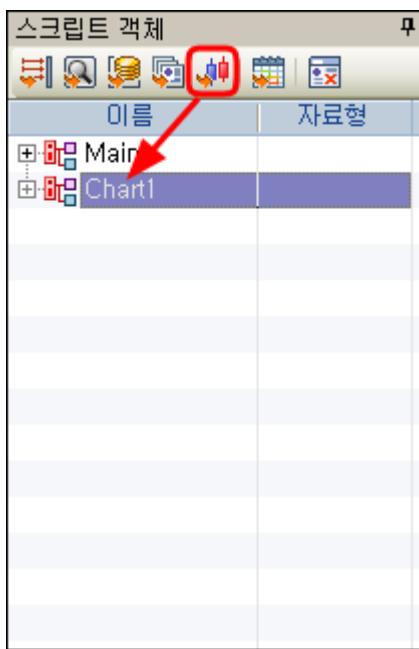
#### 2) 스크립트 객체화면 설정

선물차트에서 매수신호가 발생시 ATM 등가격 콜옵션에 주문을 하기 위해서는 첫째, 차트에서 신호발생 등의 정보를 가져오는 차트객체가 필요하고, 둘째, 현재 ATM 등가격 콜옵션 종목코드 등의 옵션정보를 확인하기 위해서는 옵션객체가 필요하며, 셋째, 주문을 위해서는 계좌객체가 필요하게 됩니다.

전략을 구현하기 위해서 먼저 필요한 차트/옵션/계좌 객체를 스크립트 객체화면에서 추가하고 스크립트 편집창에서 전략을 작성해야 합니다.

##### A. 차트객체 추가

- i. 차트에서 주는 정보를 얻기 위해서 스크립트 객체화면에서 차트객체를 추가합니다  
차트객체는 복수로 추가해서 사용 가능합니다. 예제에서는 하나의 차트객체만 필요하므로 하나만 추가합니다.



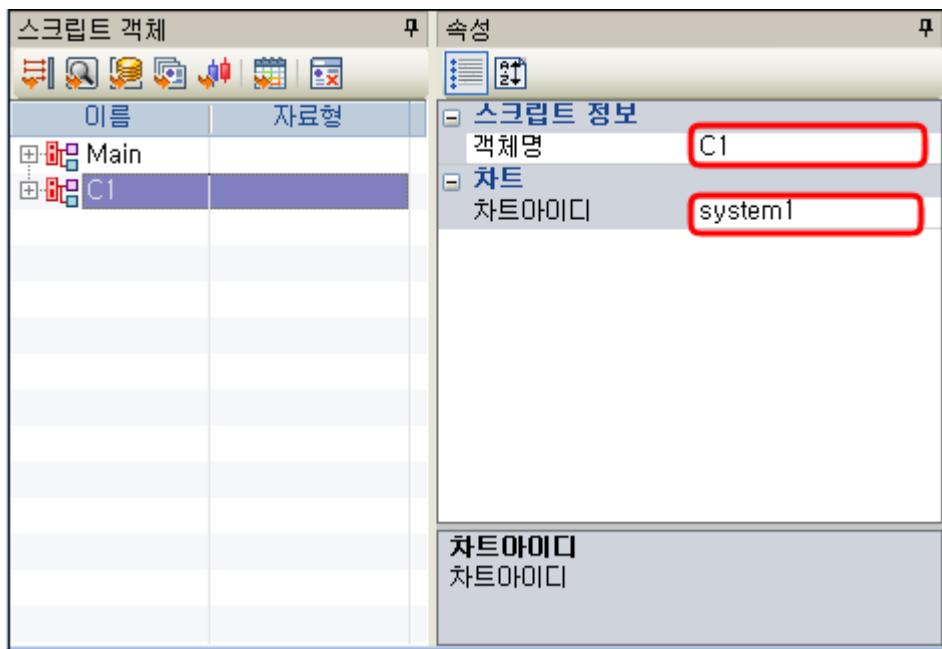
스크립트 객체화면에 차트객체를 추가한 후 속성에서  
차트객체와 연동이 되는 차트를 지정을 해야 사용할 수 있습니다.  
차트객체와 차트의 연결은 [차트아이디]를 통해서 이루어지게 되므로  
차트객체와 차트에 같은 이름으로 [차트아이디]를 꼭 부여해 주어야 합니다.

ii. 차트객체에 차트 아이디 부여

스크립트 객체화면에서 차트객체의 이름을 선택하면  
속성화면에서 해당 차트객체의 객체명과 차트아이디를 설정할 수 있습니다.

차트객체명은 기본으로 제공되는 이름을 사용해도 되고 임의로 지정해도 됩니다.

예제에서는 C1로 객체명을 변경하고 연동되는 차트아이디는 system1이라는 명칭으로 부여합니다.



차트객체명이 C1으로 변경되고 차트아이디는 system1로 지정되어

사용자가 준비한 차트 중 system1이라는 아이디를 가지는 차트에서 정보를 수신하게 됩니다.

iii. 차트에서 차트아이디 부여



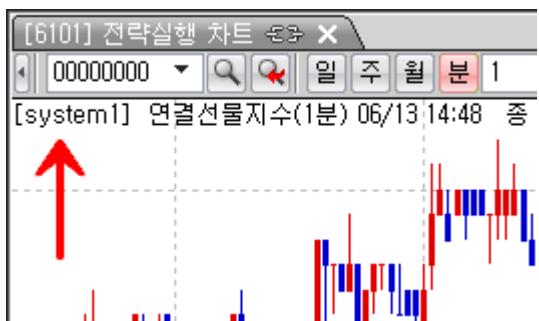
차트에서 차트아이디는 오른쪽 하단의 버튼을 클릭하면 지정할 수 있습니다.

차트객체에서 지정한 이름과 동일한 이름으로 지정해야 하며

알파벳 대소문자를 구별하므로 대소문자에 사용에 유의해야 합니다.



차트에 아이디 부여가 완료되면 차트 오른쪽 상단에 해당 차트의 아이디가 보여지게 됩니다.



차트에서 신호를 정보를 받아야 하므로 차트에 시스템식을 적용해 주면 됩니다.

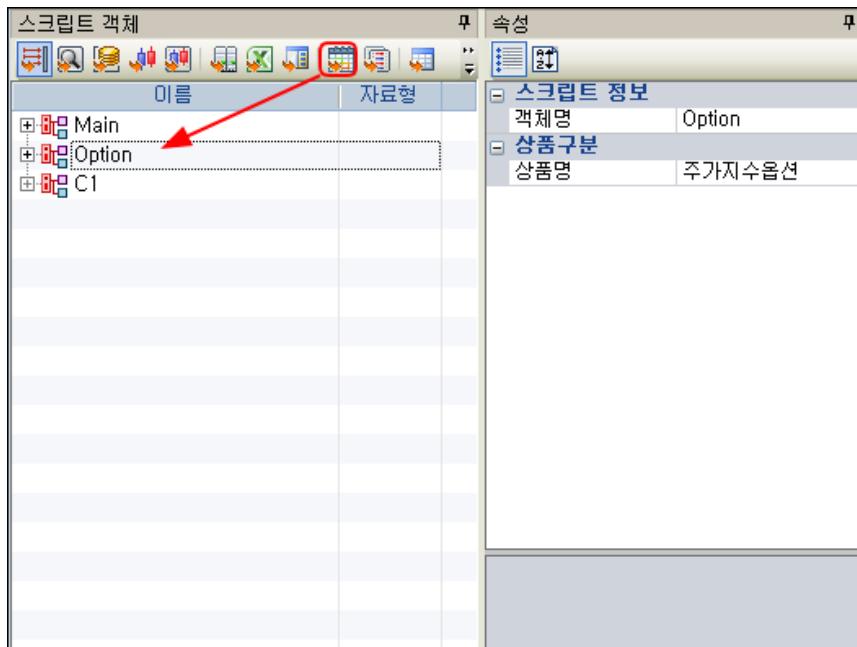
예제에서는 시스템식 중 [이동평균(단순) Golden\_Dead]를 적용했습니다.



위와 같이 설정되면 스팟전략에서 차트객체(C1)는 system1이라는 아이디를 가지는 차트에서 정보를 받게 되고 스팟전략에서 사용할 준비가 모두 마친 상태가 됩니다.

## B. 옵션객체 추가

- 실시간으로 ATM 등가격 콜옵션 정보를 확인하기 위해 옵션객체를 추가합니다.



- 옵션객체를 선택 후 속성에서 객체명은 Option으로 지정하고 상품은 주가지수옵션을 지정합니다.

C. 계좌객체 추가

- i. 주문을 낼 때 계좌번호 등이 필요하므로 계좌객체를 추가합니다.



- ii. 스크립트 객체화면에서 계좌객체명을 클릭하면

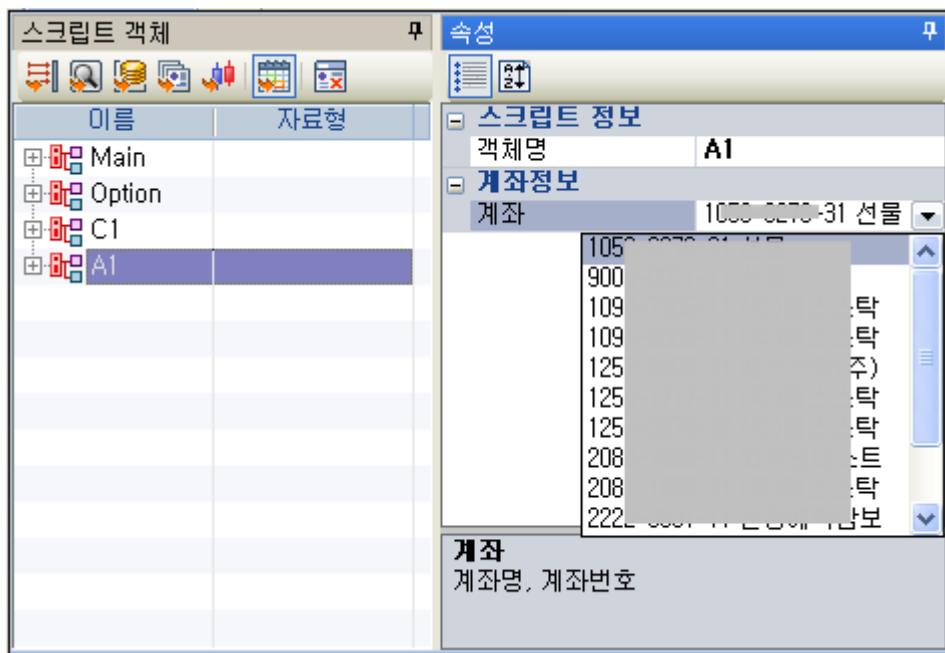
속성화면에서 해당 계좌객체의 객체명과 계좌번호를 지정할 수 있습니다.

객체명은 기본으로 지정된 이름을 사용해도 되고 임의로 지정해도 됩니다.

예제에서는 객체명은 A1로 지정합니다.

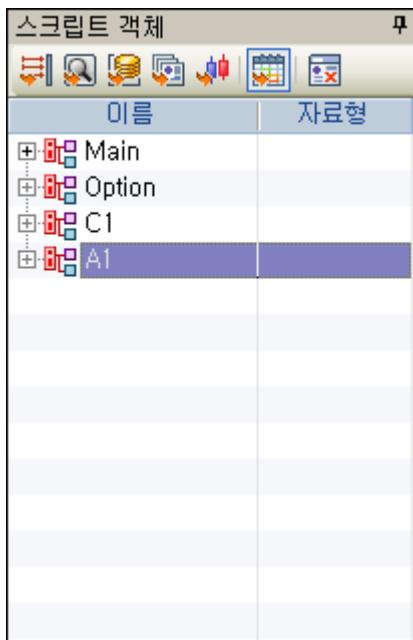
계좌란을 클릭하면 보유한 계좌의 목록이 나타납니다.

옵션종목에 주문을 해야 하므로 선물옵션 계좌번호를 지정하면 됩니다.



계좌 객체가 A1이라는 객체명으로 변경되고 지정한 계좌번호의 정보를 가지는 객체가 됩니다.

### iii. 스크립트 객체 설정 완료



### 3) 전략작성

전략 구현에 필요한 객체가 준비가 되었으므로

선물차트에서 매수신호 발생시 ATM 등가격 콜에 매수주문하는 전략을 작성해 보겠습니다.

예스스팟 전략의 기본적인 구조는

이벤트 발생(특정사건 발생)하면 그 때 처리해야 할 내용을 기술하는 것입니다.

예제1에서는 이벤트는 선물차트에서 신호 발생하는 것이며 신호가 매수신호임을 확인하는 것은 조건문, 옵션종목으로 주문을 내는 것이 실행문이 됩니다.

#### A. 이벤트 설정

먼저 선물차트에서 차트에서 완성신호가 발생하면 완성신호의 정보를 수신 받는 이벤트를 설정합니다.

이벤트는 아래와 같이 스크립트에 기술하고 중괄호 안에 처리해야 할 내용을 기술합니다.

```
function 객체명_이벤트(매개변수)
{
    처리 내용
}
```

차트에서 완성신호가 발생되면 호출되는 이벤트는 차트객체에 제공되는 OnRiseSignal이며 아래와 같이 작성해 사용합니다.

```
function C1_OnRiseSignal(Signal)
{
    |
}
```

위와 같이 작성하면 차트객체(C1)에서 완성신호이벤트(OnRiseSignal)가 발생하면 완성신호객체(Signal)에 그 정보를 넘겨주게 됩니다.

완성신호 객체인 Signal은 아래와 같은 프로퍼티를 가집니다.

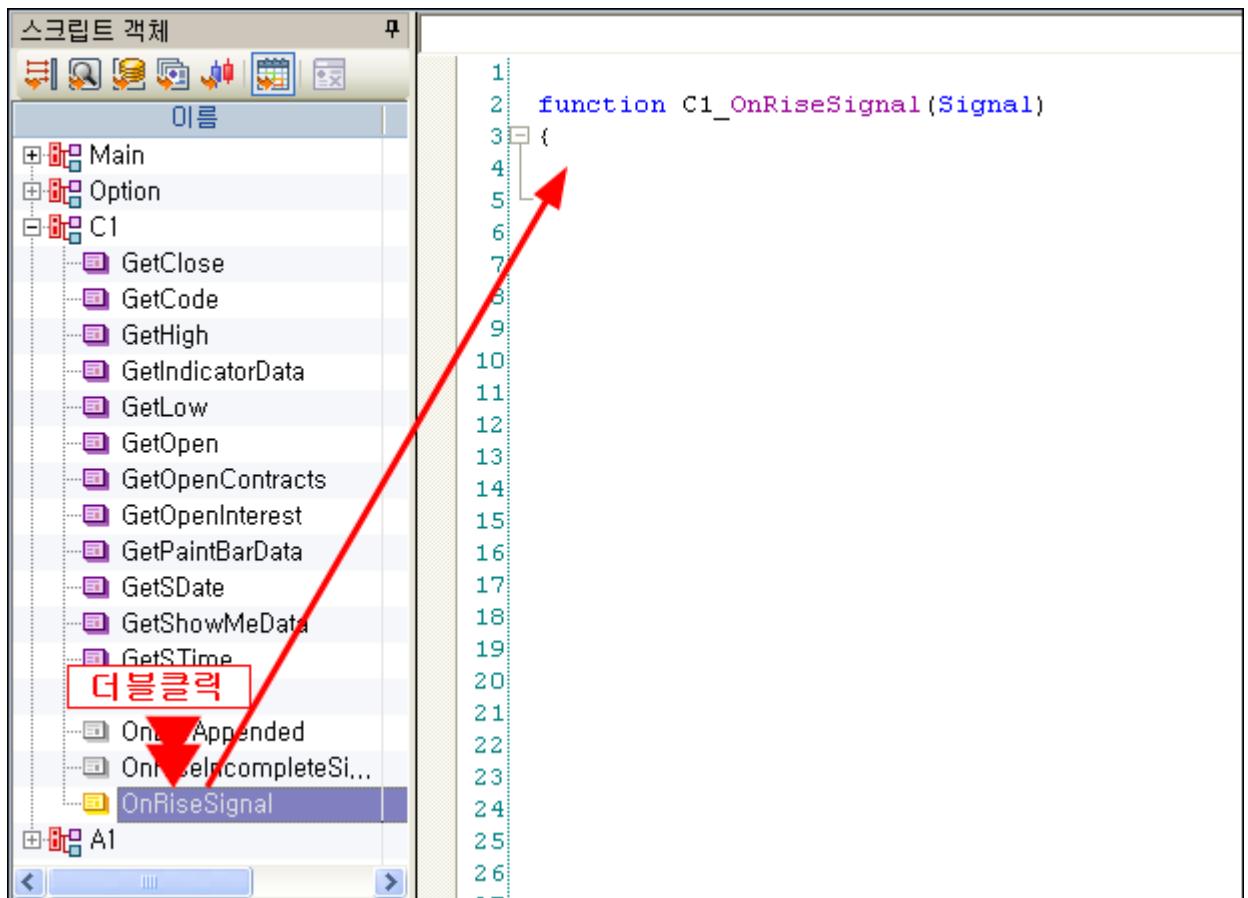
Signal	객체	신호 객체, 차트에서 발생한 신호에 대한 정보를 담고 있는 객체입니다.
apply	정수	시스템 적용구분, (0 : 시험적용, 1 : 경보 후 주문, 2 : 자동주문)
code	문자열	완성신호가 발생한 종목코드(단축코드)
count	정수	주문수량
date	정수	완성신호가 발생한 날짜
name	문자열	신호명
price	실수	신호가격
signalKind	정수	완성신호 종류(1 : Buy, 2 : ExitLong, 3 : Sell, 4 : ExitShort)
signalType	정수	완성주문신호 유형(OnClose : 1, AtMarket : 2, AtLimit : 3, AtStop : 4)
time	정수	신호시간

프로퍼티나 메소드는 이미 시스템에서 정의된 기능을 불러와 사용하는 개념으로 스크립트에서 사용할 때에는 점 연산자의 왼쪽에는 객체가, 오른쪽에는 프로퍼티나 메소드를 지정하여 사용합니다. 가령 완성신호객체(Signal)의 이름을 알고자 하면 Signal.name, 완성신호객체의 종목코드를 알고자 하면 Signal.code와 같이 표현해서 전략에서 해당 정보를 이용하면 됩니다.

※ 예스스팟에서 전략을 작성할 때에는 작성자가 하나하나 모두 기술 할 필요는 없습니다.

스크립트 객체화면에 추가되어 있는 객체의 이벤트는 모두 더블클릭하면  
스크립트 편집창에 자동으로 입력되므로 작성자가 하나하나 모두 기술 할 필요는 없습니다.

이벤트는 더블클릭하면 작성자가 스크립트에서 사용할 수 있게 스크립트 편집창의 가장 마지막 줄에  
자동으로 입력됩니다.



## B. 조건문 작성

완성신호 이벤트가 설정이 되었으므로

완성신호가 buy신호인지 확인하는 내용을 기술해 보도록 하겠습니다.

완성신호 객체에서 신호의 종류를 알 수 있는 프로퍼티는 signalKind입니다.

완성신호가 Buy이면 1, Exitlong이면 2, Sell이면 3, ExitShort이면 4를 리턴합니다.

차트에서 발생한 완성신호가 Buy이면 signalKind는 1을 가지게 되므로

아래와 같이 if문으로 완성신호의 종류가 매수임을 나타내는 if 조건문을 작성해 주시면 됩니다.

```

1  function C1_OnRiseSignal(Signal)
2  {
3      if (Signal.signalKind == 1)
4      {
5          |
6      }
7  }
8
9
10

```

※ 자바스크립트에서 if문은 if (조건) { 실행문 } 과 같은 구조이며  
알파벳 대소문자를 구별하므로 특별히 대소문자 사용에 유의하시기 바랍니다.

#### C. 실행문 작성

차트에서 발생된 신호가 Buy라는 것이 확인되면 주문을 발생하면 되므로  
if문의 실행문으로 매수주문함수가 들어가면 됩니다.

Main객체의 매수주문 함수는 아래와 같은 매개변수를 가지고  
순서대로 주문계좌번호, 주문종목코드, 주문수량, 주문가격, 가격구분입니다.

**Main.OrderBuy(sAccoutnNumber, sItemCode, nCount, dPrice, nPriceKind)**

간단히 시장가로 ATM 등가격 콜옵션을 1계약 주문한다고 하면 아래와 같이 지정해 주면 됩니다.

**Main.OrderBuy(A1.number, Option.GetATMCallRecent(0), 1, 0, 1);**

계좌번호 : A1.number, 계좌객체(A1)의 계좌번호

종목코드 : Option.GetATMCallRecent(0), 옵션객체의 ATM 등가격 콜옵션의 종목코드

주문수량 : 1

주문가격 : 시장가 이므로 0

가격구분 : 1, 시장가

※ 계좌객체에 제공되는 주문함수를 이용하면 따로 주문계좌번호를 지정하지 않아도 됩니다.

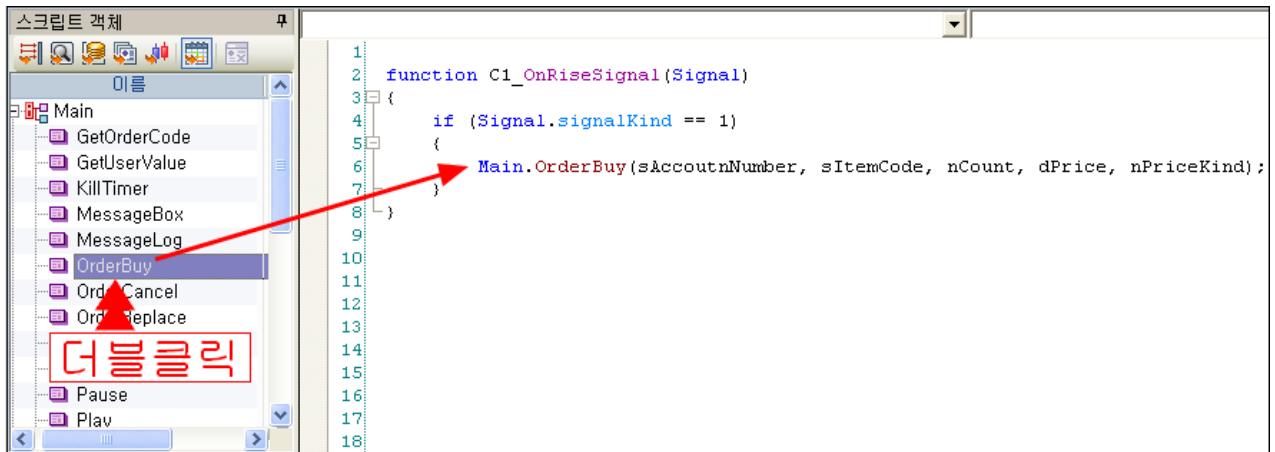
계좌객체에 제공되는 주문함수에는 해당 계좌의 계좌번호가 디폴트로 설정됩니다.

**A1.OrderBuy(Option.GetATMCallRecent(0), 1, 0, 1)**

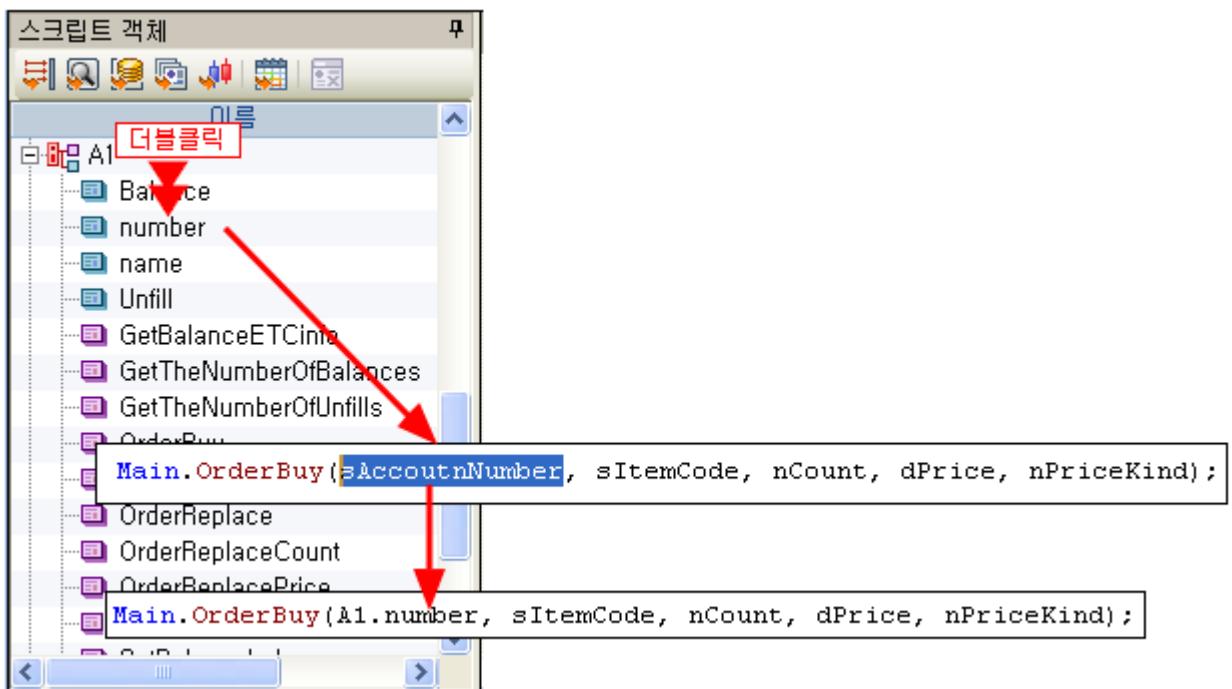
※ 스크립트에 입력될 내용 중 스크립트 객체화면에서 있는 메소드, 프로퍼티는 간단히 더블클릭하면  
자동으로 입력해 사용할 수 있습니다.

주문함수는 스크립트에서 주문함수를 추가할 위치에 커서를 위치한 후

Main 객체에 제공되는 OrderBuy를 더블클릭하면 매수주문함수를 작성자가 사용할 수 있게 자동으로 입력됩니다.



매수주문 함수를 추가한 후 계좌번호와 주문종목의 종목코드는 스크립트 객체화면에  
메소드나 프로퍼티로 제공되므로 입력될 자리에 마우스 커서를 위치하거나 블록을 설정한 후  
스크립트 객체에서 해당 객체의 메소드나 프로퍼티를 더블클릭해서 입력해 주면 됩니다.



주문수량과 주문가격, 가격구분은 직접 입력해 주셔야 합니다.

#### 4) 스팟수식

##### A. Main 객체의 주문함수 사용

/\*스크립트시작-----\*/

```

function C1_OnRiseSignal(Signal)
{
    if (Signal.signalKind == 1)
    {
        Main.OrderBuy(A1.number, Option.GetATMCallRecent(0), 1, 0, 1);
    }
}
/*스크립트끝-----*/

```

B. 계좌객체의 주문함수 사용

```

/*스크립트시작-----*/
function C1_OnRiseSignal(Signal)
{
    if (Signal.signalKind == 1)
    {
        A1.OrderBuy(Option.GetATMCallRecent(0), 1, 0, 1);
    }
}
/*스크립트끝-----*/

```

## 예제2. 복수 종목데이터 이용

2개의 종목객체를 이용한 전략으로 종목객체 사용법과 외부변수 사용법에 대한 예제입니다.

1) 전략내용

종합주가지수의 시초가가 전일 종가대비 상승하고 시초가 대비 0.05% 하락하면  
삼성전자를 매도2호가에 10주 매수

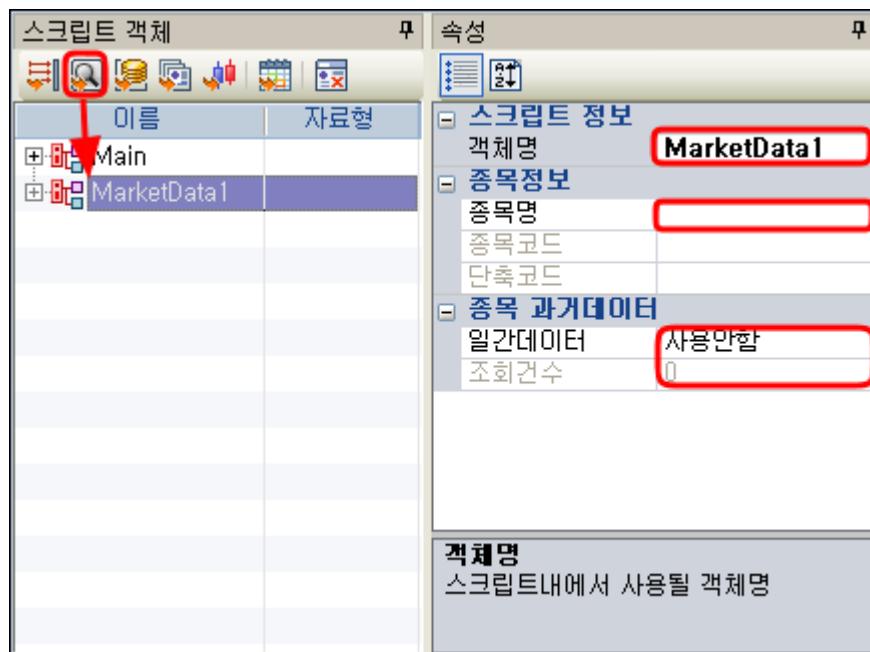
2) 스크립트 객체화면 설정

스크립트 객체설정  
종합주가지수의 가격을 이용해서 삼성전자에 주문을 해야 하므로  
종합주가지수의 종목객체, 삼성전자 종목객체가 필요하며  
주문을 위해서는 계좌객체가 필요합니다.

외부변수를 사용하여 등락률을 지정할 수 있게 지정합니다. 여기서는 초기값을 0.05로 입력합니다.

A. 첫번째 종목객체 추가

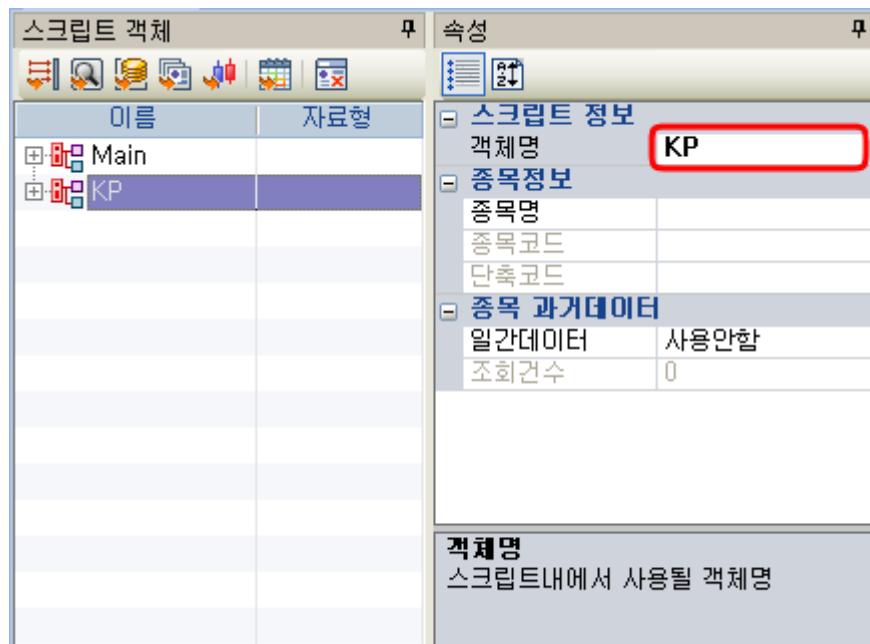
- i. 종합주가지수의 정보를 얻기 위해 첫번째 종목객체 추가합니다.



종목객체의 속성화면에서는 객체명과 종목, 일간데이터 사용건수를 지정할 수 있습니다.

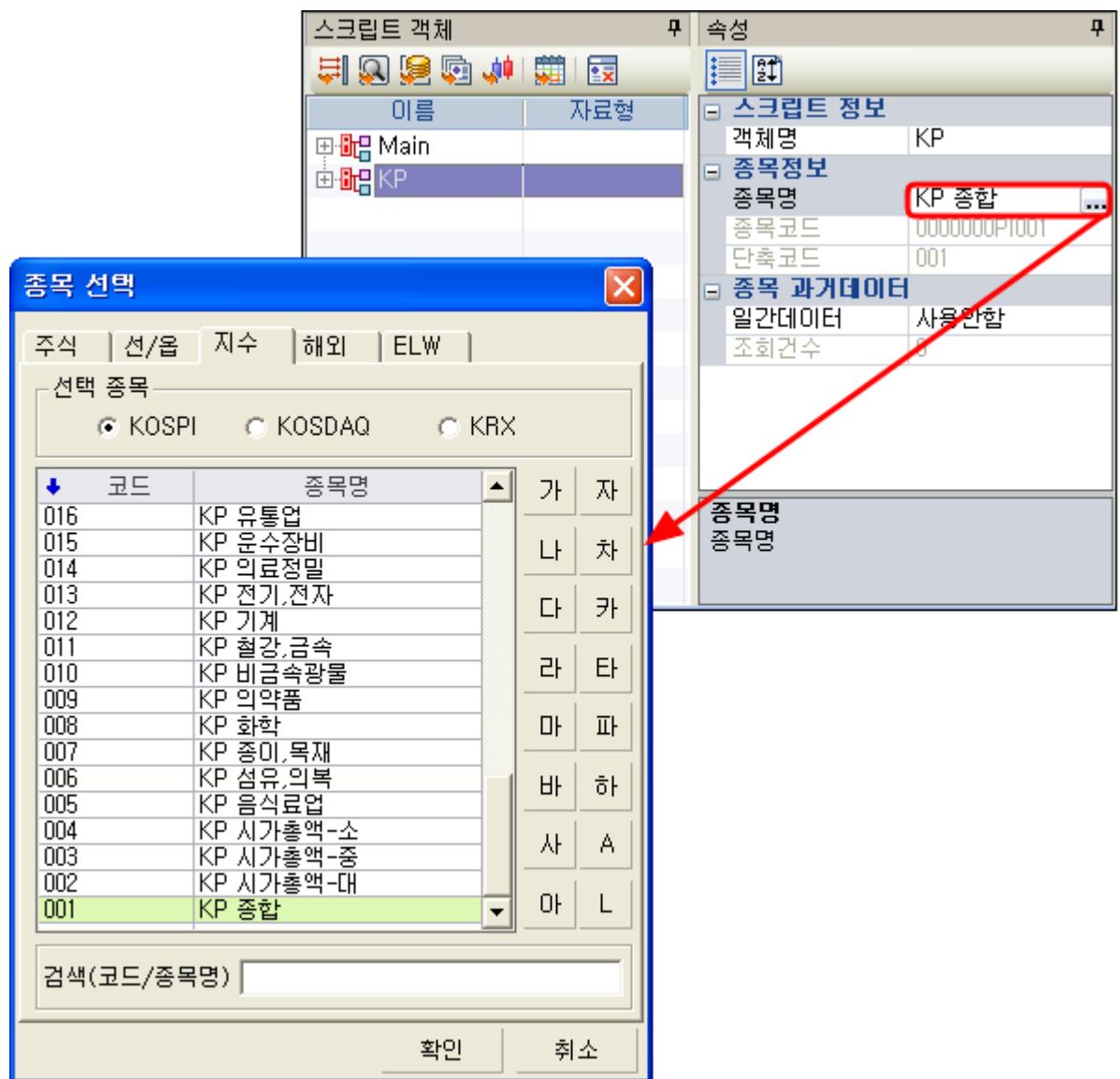
- ii. 객체명은 기본으로 제공되는 이름을 사용해도 되고 임의로 지정해도 됩니다.

예제에서는 KP로 지정해 사용하겠습니다.

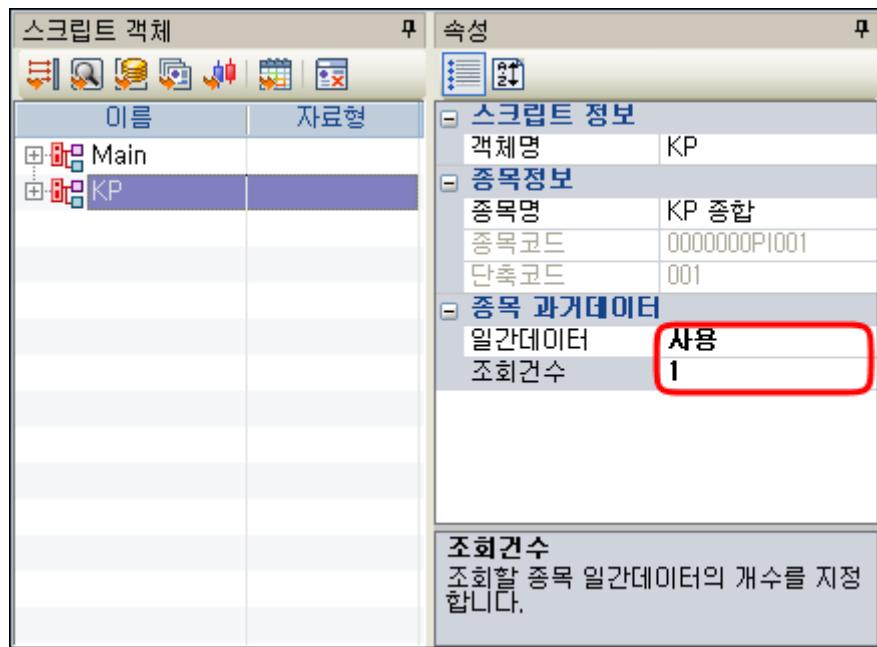


- iii. 종목명란을 누르면 종목을 지정할 수 있는 종목선택창이 나타납니다.

종목선택창에서 KP종합을 선택합니다.

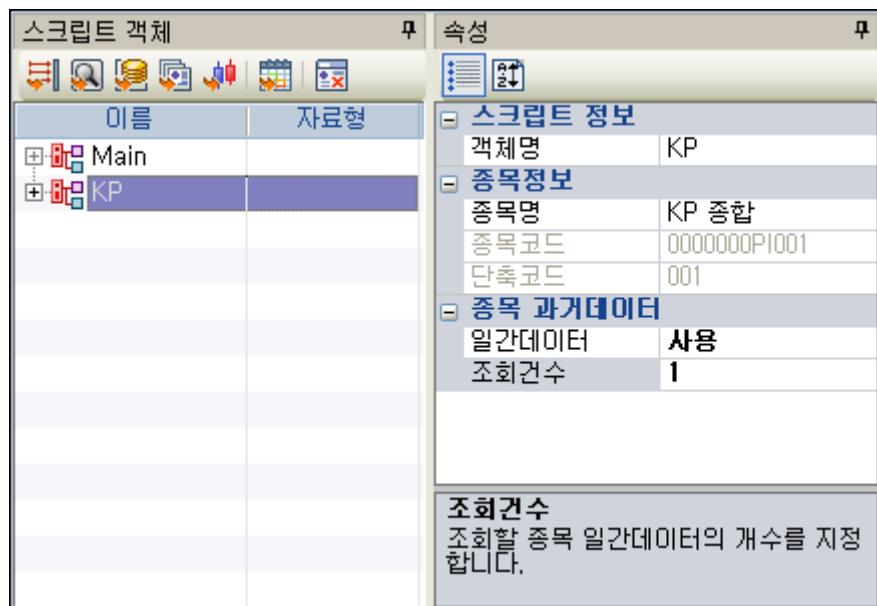


- iv. 전략에서 종합주가지수의 시초가가 전일종가대비 상승한 것을 판단해야 하므로 전일종가데이터가 필요합니다. 일간데이터는 사용으로 설정하고 전일값 한개만 필요하므로 조회건수는 1로 설정합니다.



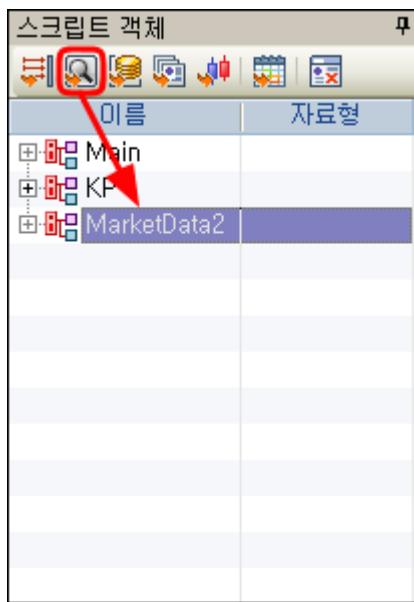
※ 일간데이터는 1~300 사이에서 지정해 사용할 수 있습니다.

#### v. 종목객체 설정 완료



#### B. 두번째 종목객체 추가

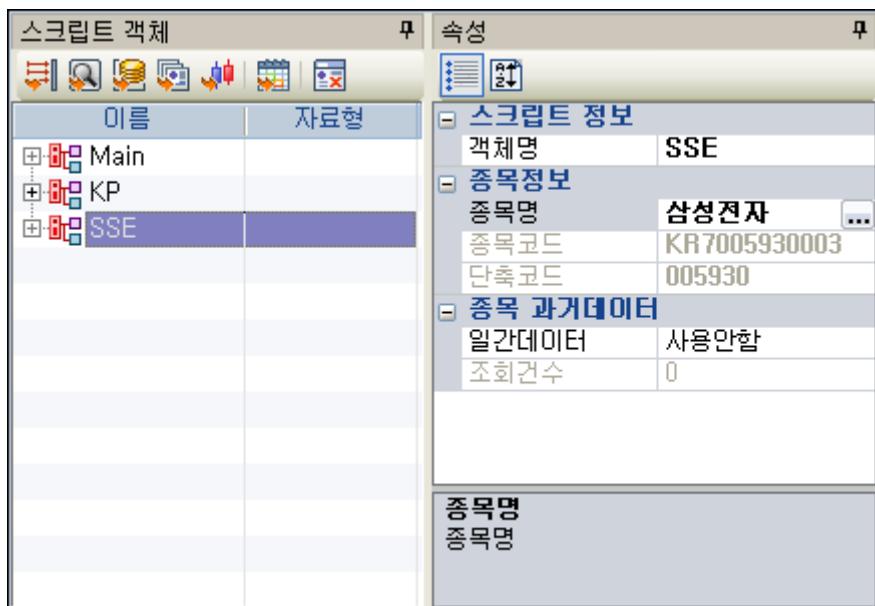
- i. 거래할 종목의 정보를 얻기 위해 두번째 종목객체를 추가합니다.



ii. 종목객체를 추가하는 방법은 첫번째와 같습니다.

객체명은 SSE, 종목은 삼성전자로 설정하고

일간데이터는 필요하지 않으므로 사용안함으로 설정합니다.

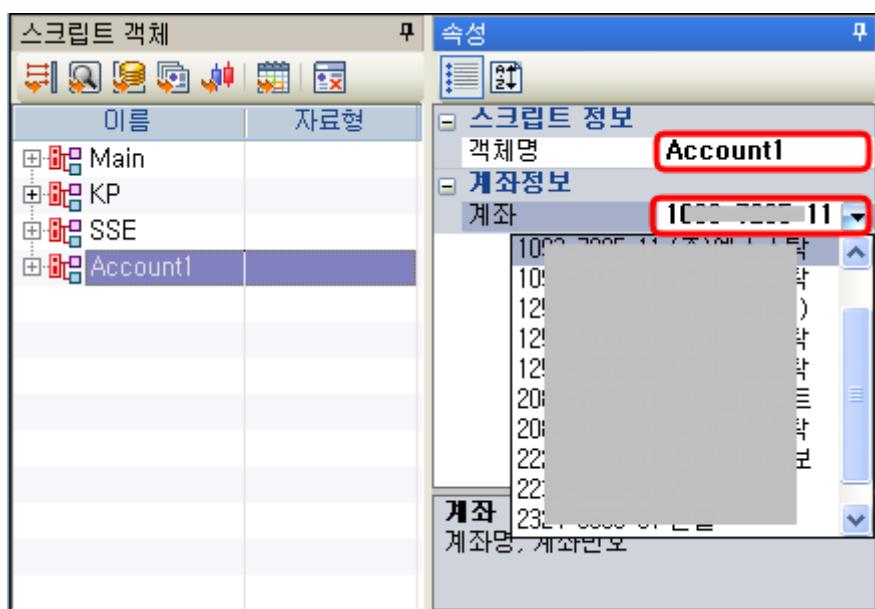


### C. 계좌객체 추가

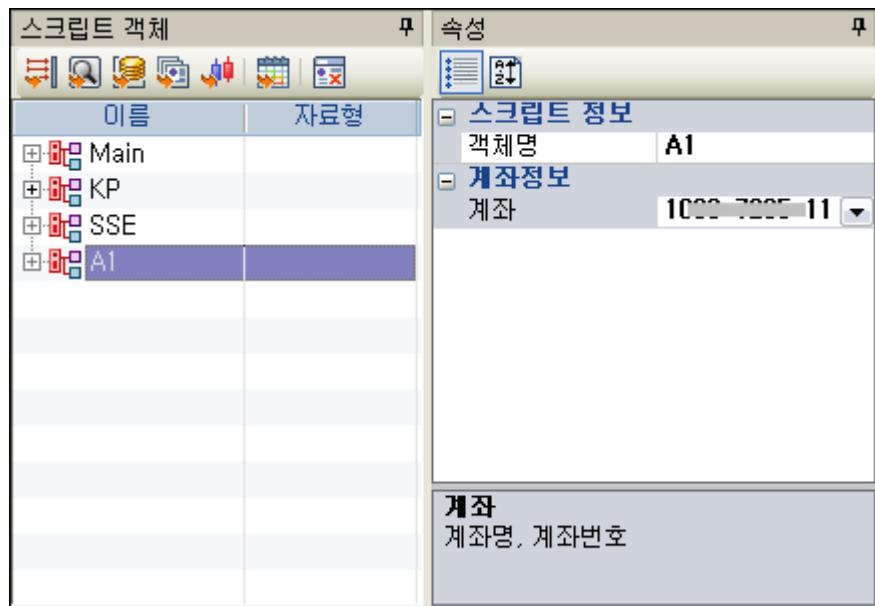
i. 거래에 사용될 계좌객체를 추가합니다.



- ii. 계좌객체를 추가한 후 속성화면에서 객체명과 계좌를 설정해 주어야 합니다.

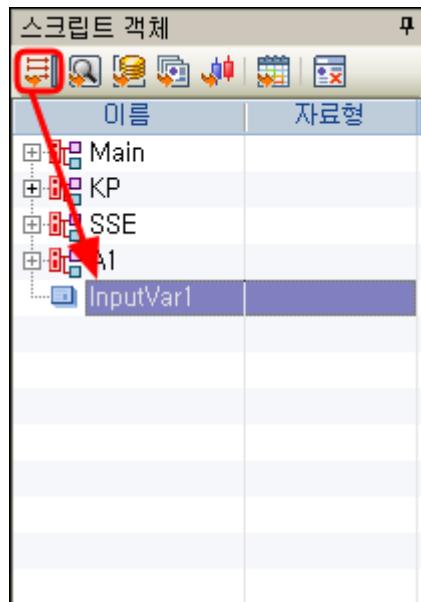


- iii. 객체명은 기본으로 제공되는 이름을 사용해도 되고 임의로 지정해 사용할 수 있습니다.  
예제에서 객체명은 A1로 지정합니다.  
계좌는 일반종목의 거래가 가능한 위탁계좌로 설정해야 됩니다.

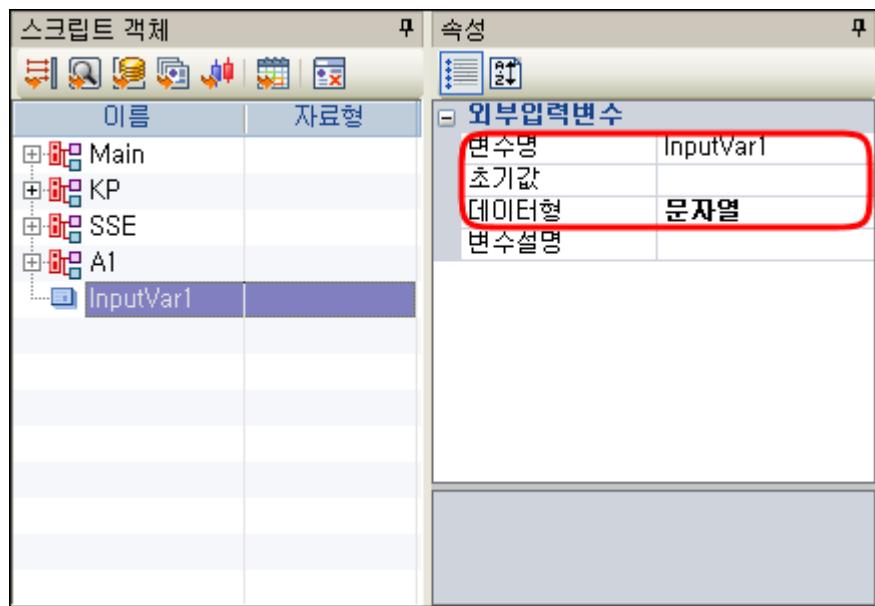


#### D. 외부변수 추가

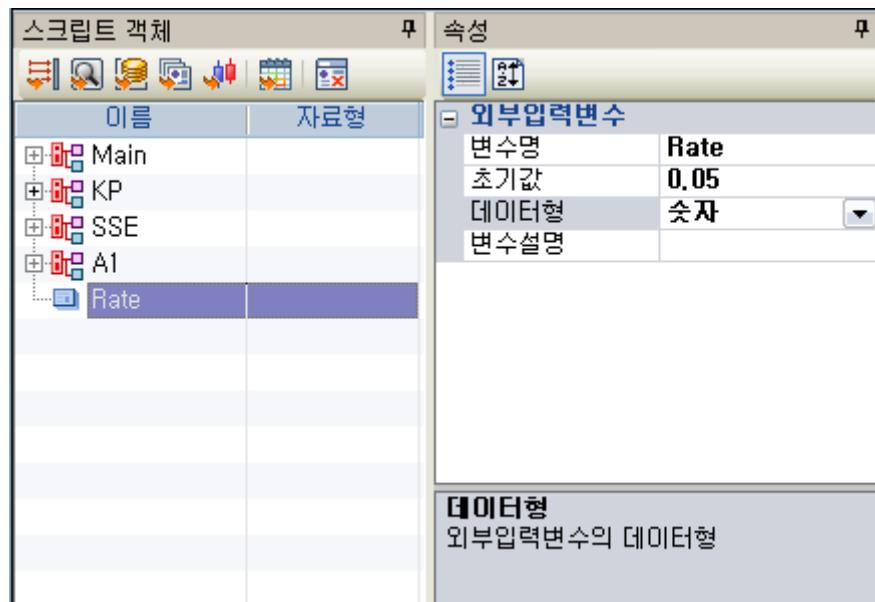
- i. 외부변수를 사용하게 되면 스팟전략을 [예스스팟 모니터]에 적용할 때  
외부변수의 기본값을 변경해서 적용할 수 있습니다.  
외부변수는 스크립트 객체화면에서 추가해서 사용할 수 있습니다.



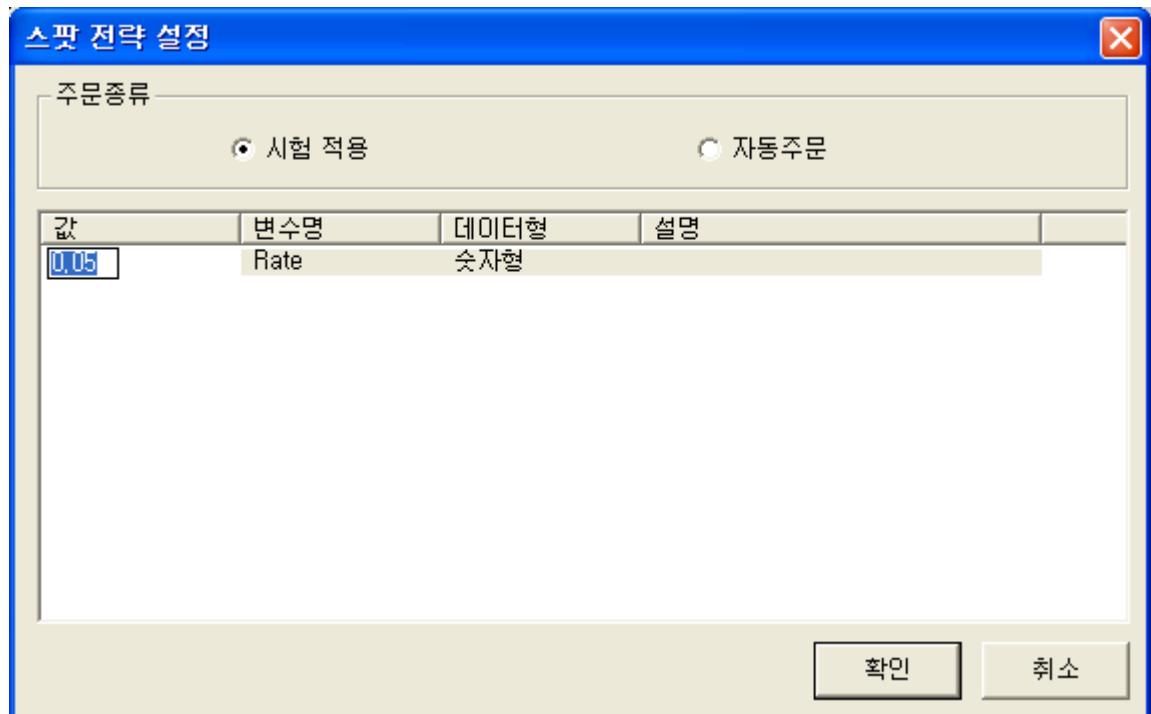
- ii. 외부변수는 추가한 후 속성화면에서 변수명, 초기값, 데이터형을 지정해 주며  
변수설명을 입력해 해당 변수의 용도를 명확히 알 수 있게 합니다.



- iii. 변수명은 기본으로 지정된 이름을 사용해도 되며 사용자가 임의로 지정해도 됩니다.  
예제에서는 Rate로 지정합니다.  
시초가 대비 하락률인 0.05를 초기값으로 지정하고  
0.05는 등락률로 사용될 수치이므로 데이터형은 숫자로 설정합니다.



외부변수를 사용하면 [예스스팟 모니터]에 전략을 적용할 때 나타나는 [스팟전략 설정]화면에 해당 변수가 표시되고 값을 변경할 수 있습니다.



### 3) 전략작성

전략 작성에 필요한 객체가 모두 준비 되었으므로

종합주가지수의 시초가가 전일 종가대비 상승하고 시초가 대비 0.05% 하락하면 삼성전자를 10주 매수하는 전략을 작성해 보도록 하겠습니다.

예제2에서는 종합주가지수의 시세 변화를 감시하는 것이 이벤트가 되고  
시초가 대비 0.05%하락하는 것이 조건문이 되며  
삼성전자를 10주 매수하는 것이 실행문이 됩니다.

#### A. 이벤트 설정

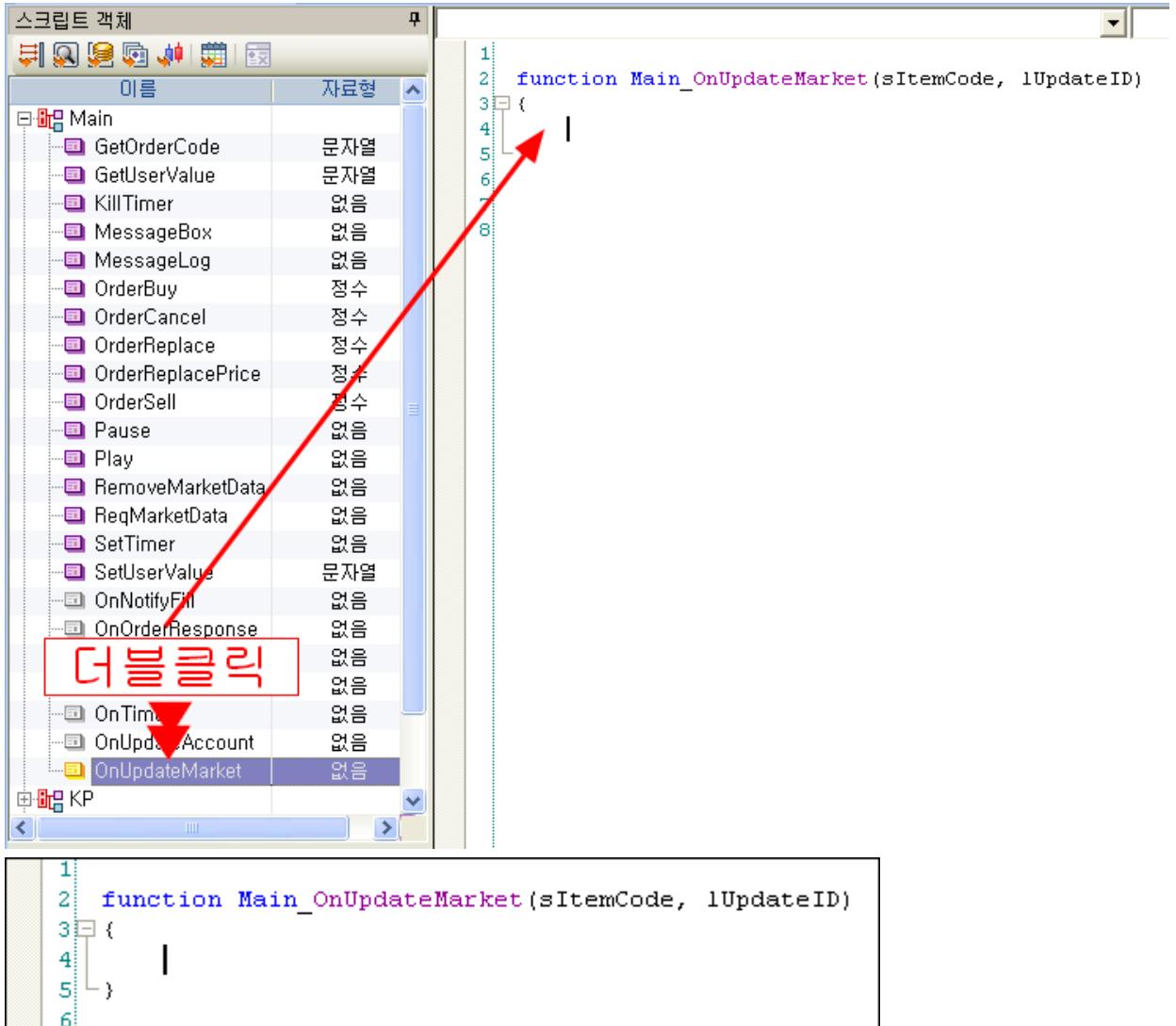
이벤트는 아래와 같이 스크립트에 기술하고 중괄호 안에 처리해야 할 내용을 기술합니다.

```
function 객체명_이벤트(매개변수)
{
    처리 내용
}
```

시세나 호가가 업데이트되면 호출되는 이벤트는 Main객체에 제공되는 OnUpdateMarket입니다.

```
function Main_OnUpdateMarket(sItemCode, lUpdateID)
{
    |
}
```

스크립트 객체화면에서 Main객체의 OnUpdateMarket을 더블클릭하면 편집창에 자동으로 입력이 되므로 직접 기술하지 않아도 됩니다.



스크립트 객체화면에 있는 종목객체 중에 시세나 호가가 변경되는 이벤트(OnUpdateMarket)가 발생하면 업데이트된 종목의 종목코드는 sItemCode로 리턴해 주고, 업데이트의 종류는 lUpdateID가 리턴해 주게 됩니다.

#### B. 조건문 작성

시세나 호가 변경에 대한 이벤트가 설정이 되었으므로 조건문을 지정해 보도록 하겠습니다.

이벤트가 발생할 때 조건문은

시세나 호가 업데이트가 발생한 종목을 확인하는 조건,  
업데이트의 종류가 시세인지 호가인지 확인하는 조건,

시초가 대비 0.05% 하락한 시세인지를 확인하는 조건.

시초가가 전일대비 상승해서 시작한 것을 확인하는 조건이 필요합니다.

또한 추가로 종합주가지수의 현재가가 시초가 대비 0.05% 아래에서 계속 형성되면 위 조건이 계속 만족하여 주문이 반복적으로 발생할 수 있으므로

처음 0.05% 아래로 내려간 시점만 주문하도록 하는 내용도 필요하게 됩니다.

i. 종합주가지수의 업데이트인지를 확인하는 조건 추가

```
1
2  function Main_OnUpdateMarket(sItemCode, lUpdateID)
3  {
4      if ( sItemCode == KP.code )
5      {
6          |
7      }
8  }
9
```

sItemCode == KP.code는 종목객체 중에 시세나 호가가 업데이트되는 이벤트가 발생했는데 그 종목의 코드는 종합주가지수의 종목코드(KP.code)와 같다라는 표현으로 현재 시세나 호가가 변경되는 이벤트가 종합주가지수에서 발생했다는 의미입니다.

ii. 업데이트가 시세 업데이트인지 호가 업데이트인지 확인하는 조건 추가

```
1
2  function Main_OnUpdateMarket(sItemCode, lUpdateID)
3  {
4      if ( sItemCode == KP.code
5          && lUpdateID == 20001 )
6      {
7          |
8      }
9  }
10
```

lUpdateID는 20001 또는 20002라는 숫자를 리턴합니다.

lUpdateID == 20001은 현재 업데이트가 시세 업데이트라는 표현입니다.

호가가 업데이트 되는 것은 lUpdateID == 20002로 표현하면 됩니다.

iii. 종합주가지수가 현재가가 시초가 대비 0.05% 이하인지를 확인하는 조건 추가

```

1  function Main_OnUpdateMarket(sItemCode, lUpdateID)
2  {
3      if ( sItemCode == KP.code
4          && lUpdateID == 20001
5          && KP.current <= KP.open*(1-Rate/100) )
6      {
7          |
8      }
9  }
10 }
11

```

KP.current <= KP.open\*(1-Rate/100)는

종합주가지수의 현재가(KP.current)가 종합주가지수의 시초가(KP.open) 대비 0.05% 이상 낮은 값이라는 표현입니다. 0.05는 외부변수를 사용합니다.

iv. 종합주가지수의 시초가가 전일종가대비 상승했다는 조건은 아래와 같이 추가

```

1  function Main_OnUpdateMarket(sItemCode, lUpdateID)
2  {
3      if ( sItemCode == KP.code
4          && lUpdateID == 20001
5          && KP.current <= KP.open*(1-Rate/100)
6          && KP.open > KP.GetPrevClose(1))
7      {
8          |
9      }
10 }
11
12

```

KP.open > KP.GetPrevClose(1)는

종합주가지수의 시초가(KP.open)가 전일종가(KP.GetPrevClose(1)) 보다 크다는 표현입니다.  
GetPrevClose은 해당 종목객체의 속성에서 과거데이터를 사용으로 설정해야 값을 리턴받는 함수입니다. 매개변수로 1은 1일전을 나타냅니다.

v. 스팟전략을 실행 후에 종합주가지수의 현재가가 시초가 대비 0.05% 이하로 처음 하락 할 때만 주문하게 하기 위해서는 아래와 같은 내용을 추가해야 합니다.

```

1 var cnt = 0;
2 function Main_OnUpdateMarket(sItemCode, lUpdateID)
3 {
4     if ( sItemCode == KP.code
5         && lUpdateID == 20001
6         && KP.current <= KP.open*(1-Rate/100)
7         && KP.open > KP.GetPrevClose(1)
8         && cnt == 0)
9     {
10        cnt = 1;
11    }
12}
13}
14}
15}

```

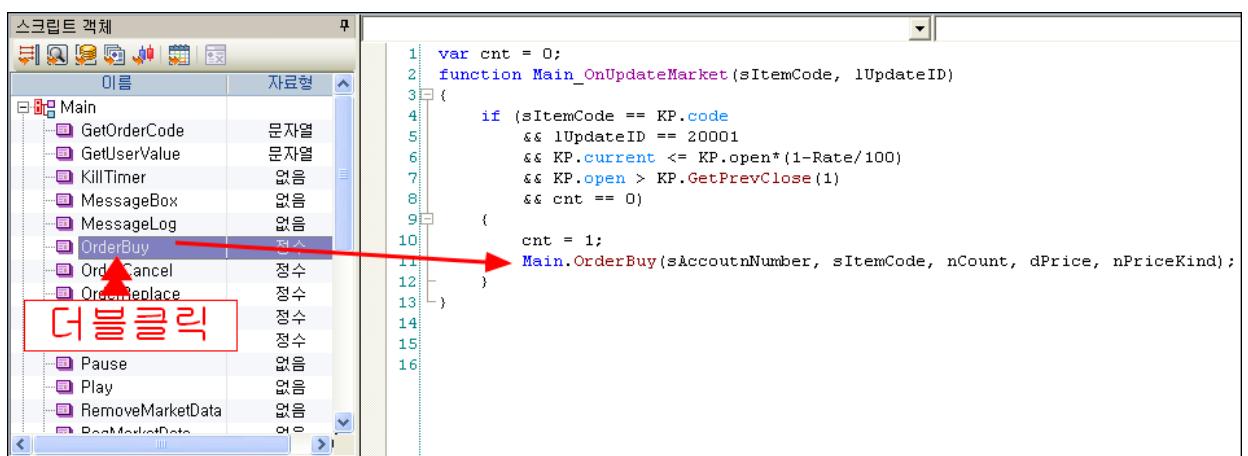
변수 cnt는 전략 실행 초기에는 0값 가지고 있다가

이벤트 발생 후 if조건에 만족하면 1로 변경되어 이후에는 if조건을 false로 만들어

더 이상 실행문이 작동하지 않습니다.

### C. 실행문 작성

이벤트와 조건문이 설정이 되었으므로 실행문으로 매수주문을 지정하면 됩니다.



Main객체에 제공되는 OrderBuy를 더블클릭하면 커서위치에 매수주문함수가 추가됩니다.

Main.OrderBuy(sAccoutnNumber, sItemCode, nCount, dPrice, nPriceKind)

매수주문함수는 주문계좌번호/주문종목코드/주문수량/주문가격/가격구분을 지정해 주어야 합니다.

삼성전자를 매도2호가로 10주 주문한다고 하면 아래와 같이 지정해 주문 됩니다.

**Main.OrderBuy(A1.name, SSE.code, 10, SSE.Ask(2), 0);**

주문계좌번호 : A1.number, 계좌객체(A1)의 계좌번호

주문종목코드 : SSE.code, 종목객체(삼성전자)의 종목코드

주문수량 : 10

주문가격 : 현재 삼성전자의 매도2호가

가격구분 : 0, 지정가

계좌객체에 제공되는 주문함수를 이용하면 따로 주문계좌번호를 지정하지 않아도 됩니다.

```
A1.OrderBuy(SSE.code, 10, SSE.Ask(2), 0);
```

#### 4) 스팟수식

##### C. Main객체의 주문함수 사용>

```
/*스크립트시작-----*/
var cnt = 0;
function Main_OnUpdateMarket(sItemCode, lUpdateID)
{
    if (sItemCode == KP.code
        && lUpdateID == 20001
        && KP.current <= KP.open*(1-Rate/100)
        && KP.open > KP.GetPrevClose(1)
        && cnt == 0)
    {
        cnt = 1;
        Main.OrderBuy(A1.number, SSE.code, 10, SSE.Ask(2), 0);
    }
}
/*스크립트끝-----*/
```

##### D. 계좌객체의 주문함수 사용>

```
/*스크립트시작-----*/
var cnt = 0;
function Main_OnUpdateMarket(sItemCode, lUpdateID)
{
    if (sItemCode == KP.code
        && lUpdateID == 20001
        && KP.current <= KP.open*(1-Rate/100)
        && KP.open > KP.GetPrevClose(1)
        && cnt == 0)
    {
}
```

```

        cnt = 1;
        A1.OrderBuy(SSE.code, 10, SSE.Ask(2), 0);
    }
}

/*스크립트끝-----*/

```

### 예제3. 시간정정주문

예제3에서는 시간정정주문을 구현하는 방법을 통해 주문응답이벤트, 체결이벤트, 타이머이벤트 등 몇 가지 이벤트에 대해 알아보도록 하겠습니다.

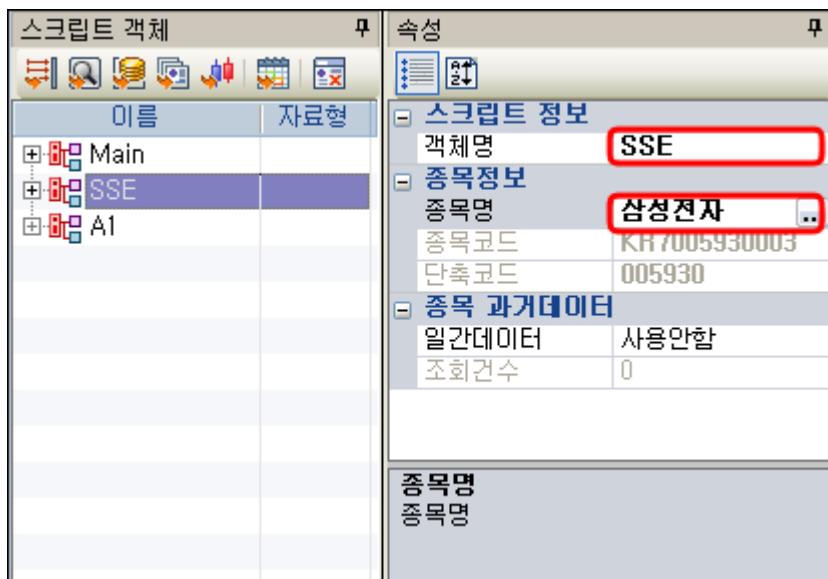
#### 1) 전략내용

스팟전략이 [예스스팟 모니터]에 적용하면  
삼성전자를 매수1호가에 1주 매수주문하고  
해당 주문에 대한 주문응답이 수신된 후 60초 후에도 미체결 상태이면  
매도2호가로 정정주문을 발생합니다.

#### 2) 스크립트 객체화면 설정

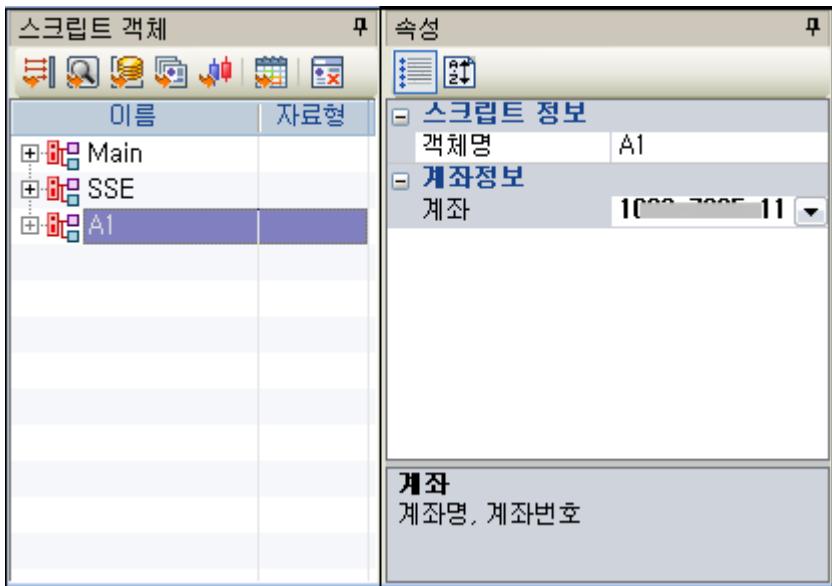
##### A. 종목객체

삼성전자에 주문을 발생해야 하므로 종목객체를 추가한 후  
속성화면에서 종목은 삼성전자로 설정하고 객체명은 F1로 지정합니다.



##### B. 계좌객체

주문을 발생해야 하므로 계좌객체를 추가한 후 계좌를 지정합니다.  
객체명은 A1로 합니다.



C. 스크립트 객체설정 완료



3) 전략작성

- A. 우선 스팟전략이 [예스스팟 모니터]에 적용되어 처음 실행될 때 매수주문을 발생해야 합니다.

```

1 var ID;
2
3 function Main_OnStart()
4 {
5     ID = A1.OrderBuy(SSE.code, 1, SSE.Bid(1), 0);
6 }
7

```

Main객체의 OnStart 이벤트는 스팟전략이 처음 실행될 때를 호출되는 이벤트입니다.

전략상 스팟전략이 처음 실행될 때, 계산해야 할 값이 있거나 초기값을 지정해 주는 등 실행해야 할 내용

이 있을 때 사용하는 이벤트입니다.

주문함수는 Main객체에도 제공되지만 Main객체의 주문함수는 계좌번호를 지정해 주어야 하므로 계좌번호를 지정할 필요가 없는 계좌객체의 주문함수를 이용합니다.

```
ID = A1.OrderBuy(SSE.code, 1, SSE.Bid(1), 0);
```

스팟에서 주문이 실행되면 모두 고유의 주문식별번호가 부여됩니다. 주문식별번호를 사용하기 위해서는 주문함수를 변수에 할당하면 자동으로 해당 주문의 주문식별번호가 변수에 저장이 됩니다. 주문식별번호는 내부적으로 미리 지정한 양식에 따라 부여되므로 항상 변수에 저장하고 사용해야 합니다. 주문식별번호를 이용하면 수신된 주문응답이 현재 전략에서 발생한 주문의 응답인지를 더 쉽게 파악할 수 있습니다. 주문식별번호를 저장한 ID변수는 다른 이벤트에서도 사용해야 하므로 전역변수로 선언합니다.

- B. OnStart이벤트로 주문이 설정이 되었으므로 주문 발생 후에 주문응답을 받으면 정정주문을 위해 주문번호를 저장하고 정정주문을 위해 시간을 셋팅합니다.

```
1  var ID;
2  var Number;
3
4  function Main_OnStart()
5  {
6      ID = A1.OrderBuy(SSE.code, 1, SSE.Bid(1), 0);
7  }
8
9  function Main_OnOrderResponse(OrderResponse)
10 {
11     if (OrderResponse.orderID == ID)
12     {
13         Number = OrderResponse.orderNum;
14         Main.SetTimer(1, 60000);
15     }
16 }
17
```

Main객체의 OnOrderResponse 이벤트는 주문응답이 발생하면 호출되는 이벤트입니다.

OnOrderResponse 이벤트는 스팟전략에서 발생된 주문에 대한 응답이 발생했을 때만 호출됩니다. 다른 주문화면에서 발생된 주문에 대한 응답에 대해서는 호출되지 않습니다.

발생한 주문응답에 대한 정보는 주문응답객체(OrderResponse)로 확인합니다. 주문응답객체가 제공하는 정보는 아래와 같습니다.

 OrderResponse	객체	주문접수응답 객체
 accountNum	문자열	계좌번호
 code	문자열	종목코드(단축코드)
 error	문자열	에러메세지
 orderCondition	정수	주문조건(IOC, FOK, ...)
 orderCount	정수	주문수량
 orderID	정수	주문식별번호, Buy나 Sell 할수를 사용했을 때 반환되는 값.
 orderKind	정수	주문구분(1 : 매도, 2 : 매수)
 orderNum	문자열	주문번호
 orderPrice	실수	주문가격
 orgOrderNum	문자열	원주문번호
 priceKind	정수	가격구분(0 : 지정가, 1 : 시장가, 2 : 최유리지정가, 3 : 지정가(IOC))

주문응답 이벤트가 발생하면 해당 주문응답이 현재 전략에서 발생한 주문에 대한 응답인지 조건문으로 확인해야 합니다.

```
function Main_OnOrderResponse(OrderResponse)
{
    if (OrderResponse.orderID == ID)
    {
        Number = OrderResponse.orderNum;
        Main.SetTimer(1, 60000);
    }
}
```

주문응답객체의 주문식별번호를 이용해 현재 수신된 주문응답의 주문식별번호가 현재 전략이 매수주문 시에 저장된 주문식별번호와 같은지 확인합니다.

동일하면 변수에 주문응답객체의 주문번호(OrderResponse.orderNum)를 저장하고 타이머를 실행합니다.

주문번호를 저장하는 변수는 다른 이벤트에서도 사용되므로 전역변수로 선언합니다.

Main객체의 SetTimer는 타이머를 설정하는 함수입니다.

수식에서 여러 개의 타이머를 사용할 수 있으므로 타이머의 아이디를 지정하고 반복주기를 설정합니다.

타이머의 아이디는 1로 지정하고 반복주기는 60초로 지정합니다. 타이머의 반복주기는 1/1000초단위로 지정이 되므로 1000이 1초이며 60000으로 지정하면 1분(60초)이 됩니다.

- C. 주문응답 이벤트가 발생하면 타이머가 설정이 되고 지정한 60초가 경과한 후 미체결이면 정정주문을 해야 합니다. 정정주문에 대한 주문응답을 받으면 다시 타이머가 작동되고 60초 후에 다시 정정주문을 발생합니다.

```

1  var ID;
2  var Number;
3
4  function Main_OnStart()
5  {
6      ID = A1.OrderBuy(SSE.code, 1, SSE.Bid(1), 0);
7  }
8
9  function Main_OnOrderResponse(OrderResponse)
10 {
11     if (OrderResponse.orderID == ID)
12     {
13         Number = OrderResponse.orderNum;
14         Main.SetTimer(1, 60000);
15     }
16 }
17
18 function Main_OnTimer(nEventID)
19 {
20     if (nEventID == 1)
21     {
22         A1.OrderReplacePrice(Number, SSE.Ask(2));
23         Main.KillTimer(1);
24     }
25 }
26

```

Main 객체의 OnTimer는 타이머가 실행되면 호출되는 이벤트이며 실행된 타이머의 아이디는 nEventID로 알 수 있습니다.

```

if (nEventID == 1)
{
    A1.OrderReplacePrice(Number, SSE.Ask(2));
    Main.KillTimer(1);
}

```

타이머를 설정할 때 아이디를 1로 지정했으므로 현재 실행된 타이머의 아이디가 1인지 확인합니다. 아이디가 동일하면 정정주문을 실행하고 아이디가 1인 타이머를 종지합니다.

정정주문함수는 계좌번호가 디폴트로 적용되어 있는 계좌객체의 정정주문함수를 이용하고 가격만 변경하므로 가격정정주문함수를 이용합니다.  
매개변수로 주문번호와 정정주문가격을 지정하시면 됩니다.

※ 정정주문 함수는 2가지 종류가 제공되고 있습니다.  
수량과 가격을 같이 정정할 경우에는 OrderReplace를 이용합니다.  
OrderReplace(sAccoutnNumber, sOrderNumber, nCount, dPrice)  
매개변수는 순서대로 계좌번호, 주문번호, 정정수량, 정정가격을 지정하시면 됩니다.

가격만 정정할 경우에는 OrderReplacePrice를 이용합니다.

OrderReplacePrice(sAccountrNumber, sOrderNumber, dPrice)

매개변수는 순서대로 계좌번호, 주문번호, 정정가격을 지정하시면 됩니다.

계좌객체의 정정주문함수를 이용하면 계좌번호는 생략됩니다.

- D. 주문이 체결이 되면 타이머는 중지됩니다.

```
27 function Main_OnNotifyFill(NotifyFill)
28 {
29     if (NotifyFill.orderNum == Number)
30         Main.KillTimer(1);
31 }
```

4) 스팟수식

```
/*스크립트시작-----*/
var ID;
var Number;

function Main_OnStart()
{
    ID = A1.OrderBuy(SSE.code, 1, SSE.Bid(1), 0);
}

function Main_OnOrderResponse(OrderResponse)
{
    if (OrderResponse.orderID == ID)
    {
        Number = OrderResponse.orderNum;
        Main.SetTimer(1, 60000);
    }
}

function Main_OnTimer(nEventID)
{
    if (nEventID == 1)
    {
        A1.OrderReplacePrice(Number,SSE.Ask(2));
        Main.KillTimer(1);
    }
}
```

```

        }
    }

function Main_OnNotifyFill(NotifyFill)
{
    if (NotifyFill.orderNum == Number)
        Main.KillTimer(1);
}

```

/\*스크립트끝-----\*/

#### **예제4. 취소주문**

예스랭귀지로 작성된 시스템은 조건이 만족하면 주문까지만을 담당하고 계좌와 연계되지 않아 미체결 여부를 알 수가 없습니다.

그러므로 Buy신호가 발생해서 매수1계약을 주문하고 현재 미체결된 상태에서 ExitLong신호가 발생하면 잔고에 매수포지션이 없으므로 청산주문이 체결이 되면 매도1계약을 가지게 됩니다.

아래 예제는 청산신호 발생하면 진입주문이 미체결이면 주문을 취소하고 체결이 되어 있으면 청산주문도 발생하게 하는 내용입니다.

##### **1) 전략내용**

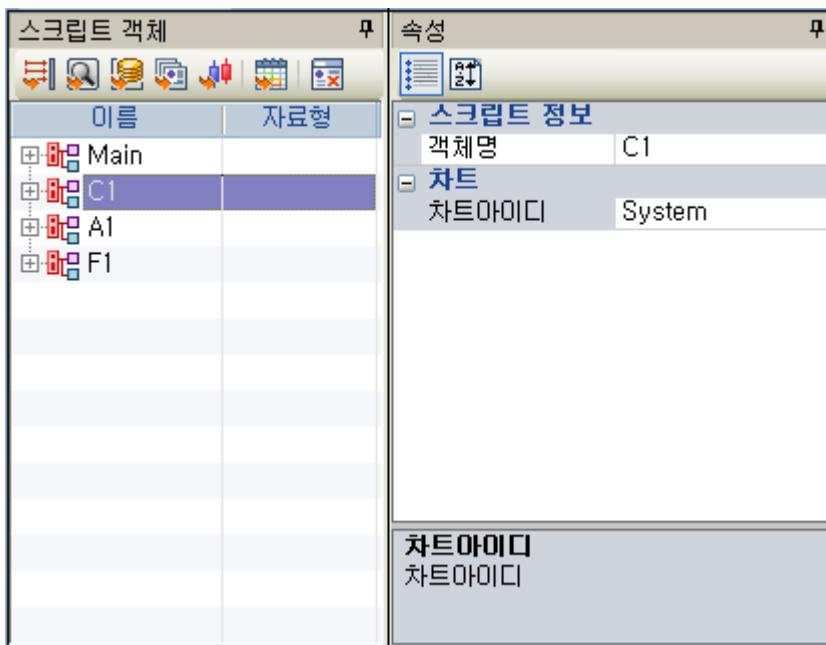
- A. Buy진입주문이 미체결이면 ExitLong신호 발생할 때 Buy신호주문을 취소하고 청산주문을 내지 않고 체결되었으면 정상적으로 주문을 집행합니다.
- B. Sell진입주문이 미체결이면 ExitShort신호 발생할 때 Sell신호주문을 취소하고 청산주문을 내지 않고 체결되었으면 정상적으로 주문을 집행합니다.

##### **2) 스크립트 객체화면 설정**

###### **A. 차트객체**

차트에 적용된 시스템 신호정보를 알아야 하므로 차트객체가 필요합니다.

예제에서는 객체명을 C1으로 하고 차트아이디는 System으로 합니다.



선물시스템이 적용된 차트에 아이디를 System으로 지정하셔야 해야 차트객체가 해당 차트에서 신호가 발생한 것을 알 수 있습니다.

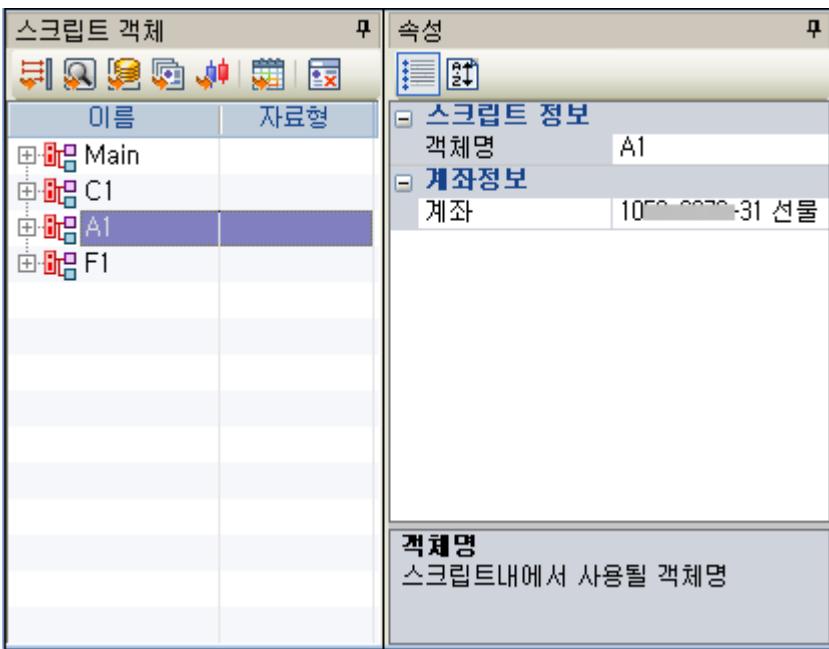


차트에서 직접 주문이 발생하는 것이 아니므로 시스템은 시험적용으로 하시면 됩니다.

#### B. 계좌객체

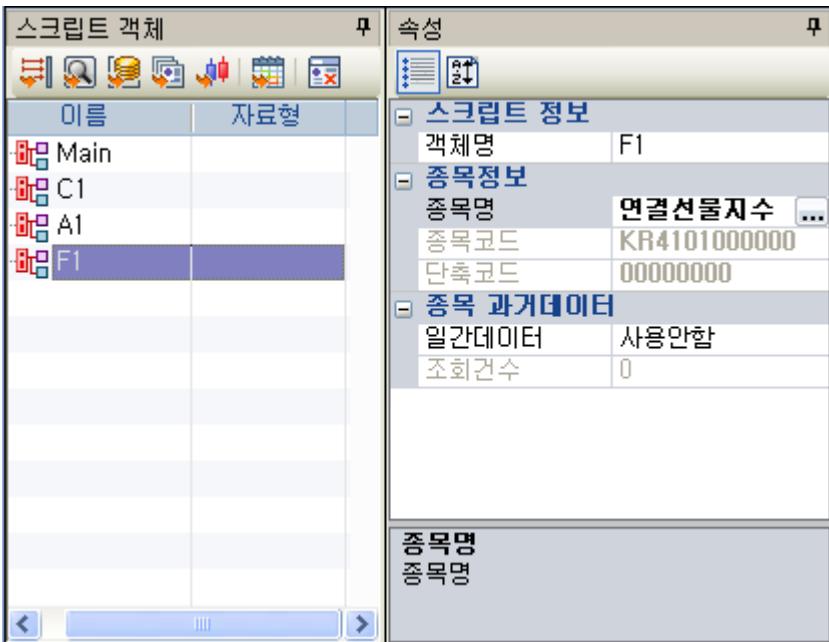
주문을 위해 계좌객체가 필요합니다.

예제에서는 객체명을 A1으로 지정하고 계좌는 선물옵션계좌로 설정합니다.



### C. 종목객체

주문을 낼 종목을 지정해야 하므로 차트의 종목과 동일 종목으로 종목객체를 설정합니다.



### 3) 전략작성

#### A. 스팟전략이 최초 실행될 때.

```

10 function Main_OnStart()
11 {
12     ① Main.MessageLog("시작");
13     ② Position = 0;
14     ③ OrderCode = Main.GetOrderCode(F1.code);
15 }
16

```

- ① [예스스팟 스튜디오]의 디버깅창과 [예스스팟 모니터]에 시작이라는 텍스트를 출력합니다.
- ② 스팟전략 실행 후에 차트에서 발생하는 최초의 진입신호부터 주문을 하기 위한 처리가 필요하므로 변수(Position)를 만들어 우선 기초값을 0으로 셋팅합니다. 전역변수로 선언합니다.
- ③ 종목객체의 종목코드를 주문용 종목코드로 리턴받아 변수(OrderCode)에 저장합니다.  
OrderCode는 전역변수로 선언합니다.
- \* Main객체의 GetOrderCode함수는 입력한 종목코드를 주문용 종목코드로 리턴해 주는 함수입니다.  
일반적으로 차트등에서 많이 사용하는 연결선물지수는 선물근월물을 연결한 데이터이므로  
종목코드인 00000000은 실제 존재하는 종목의 코드가 아니라 데이터 구분용 코드로 보시면 됩니다.  
이런 종목코드를 주문함수에 이용하면 에러가 발생하므로 연결선물지수나 ATM연결콜풋과 같은  
데이터를 종목객체나 차트의 종목으로 사용한다면 주문용 종목코드로 변경해서 주문함수에서  
이용해야 합니다.

#### B. 차트 신호와 연계된 전략이므로 차트객체(C1)의 OnRiseSignal 이벤트를 사용합니다.

```

17 function C1_OnRiseSignal(Signal)
18 {
19     Main.MessageLog("신호완성/" + Signal.signalKind);
20 }
21

```

OnRiseSignal은 연결된 차트에서 완성신호가 발생했을 때 호출되는 이벤트이고 완성신호 정보는 완성신호객체(Signal)가 제공합니다.

메시지로그 함수로 완성신호 이벤트가 발생하면 신호완성이라는 텍스트를 함께 신호종류를 디버깅창과 [예스스팟모니터]에 출력합니다.

완성신호 이벤트 발생할 때 신호종류를 확인하고 주문을 발생하거나 취소주문을 발생합니다.

#### C. Buy 완성신호가 발생했을 경우로 완성신호 이벤트 안에 기술하는 내용입니다.

```

if (Signal.signalKind == 1 )
{
    ① Position = 1;
    ② BID = A1.OrderBuy(OrderCode, Signal.count, F1.current, 0);
    ③ Main.MessageLog("매수진입");
    ④ BuyFill = 0;
}

```

- ① Position에 1값을 저장합니다.

※ 변수(Position)은 2가지 용도로 사용하기 위해 만든 변수입니다.

첫번째는 스팟전략 실행 후 최초로 발생하는 진입주문부터 주문을 발생하기 위한 용도이며 두번째는 주문응답이벤트나 체결통보이벤트 발생 시 최근 주문의 상태를 확인하는 용도입니다. 전역변수로 선언하고 사용합니다.

② 현재가로 매수주문을 발생하고 주문식별번호는 BID변수에 저장합니다.

※ OrderCode를 주문종목으로 지정

※ 주문수량으로 지정한 Signal.count는 차트시스템에서 설정된 수량입니다.

※ 저장된 주문식별번호는 주문응답 이벤트 발생 시 현재 주문의 응답인지 확인하기 위해 사용됩니다.

③ 매수진입이라는 텍스트를 디버깅창과 [예스스팟모니터]에 출력합니다.

④ BuyFill에 0을 할당합니다.

※ 변수(BuyFill)은 Buy신호시 발생한 매수주문이 체결되면 1로 변경됩니다.

D. ExitLong 완성신호 발생했을 경우로 완성신호 이벤트 발생시 기술하는 내용입니다.

```
if (Position == 1 && Signal.signalKind == 2 )
{
    ① if (BuyFill == 1)
    {
        ② A1.OrderSell(OrderCode, Vol, F1.Bid(2), 0);
        ③ Main.MessageLog("매수청산실행");
    }
    ④ if (BuyFill == 0)
    {
        ⑤ A1.OrderCancel(BNum);
        ⑥ Main.MessageLog("매수청산취소");
    }
}
```

if (Position== 1 and Signal.signalKind == 2) 조건은

완성신호의 종류가 ExitLong이고 Position에 1이 이미 저장된 상태라는 내용입니다.

스팟전략 적용하고 Buy나 Sell신호가 발생해서 Position가 1값을 가진 이후에 발생하게 한 내용입니다.

ExitLong신호가 발생할 때는 진입주문이 미체결된 경우와 체결된 경우로 나누어 조건문을 만듭니다.

①. BuyFill은 매수진입주문 시에 0으로 셋팅되고 체결통보를 받으면 1로 변경이 되게 작성되어 있습니다.

그러므로 BuyFill이 1이라는 의미는 매수진입주문이 체결된 상태를 나타냅니다.

②. 매수청산을 위해 매도주문을 집행합니다.

※ OrderCode를 주문종목으로 지정

※ 주문수량으로 지정한 Signal.count는 차트시스템에서 설정된 수량입니다.

※ 매수2호가로 지정가 매도주문합니다.

③. 매수청산실행이라는 텍스트를 디버깅창과 [예스스팟모니터]에 출력합니다.

④. BuyFill이 0이라는 의미는 매수진입이 미체결된 상태를 나타냅니다.

⑤. 매수진입주문이 체결되지 않았으므로 매수청산을 위한 매도주문은 생략하고

미체결된 매수진입주문에 대해 취소주문을 집행합니다.

※ 주문번호는 주문응답 이벤트 발생 대 저장한 Bnum입니다.

- ⑥. 매수청산취소라는 텍스트를 디버깅창과 [예스스팟모니터]에 출력합니다.

- E. Sell 완성신호 발생했을 경우로 완성신호 이벤트 발생시 기술하는 내용입니다.

```
if (Signal.signalKind == 3)
{
    ① Position = -1;
    ② SellFill = 0;
    ③ SID = A1.OrderSell(OrdeCode, Signal.count, F1.current, 0);
    ④ Main.MessageLog("매도진입");
}
```

- ① Position에 -1값을 저장합니다.

※ 변수(Position)은 2가지 용도로 사용하기 위해 만든 변수입니다.

첫번째는 스팟전략 실행 후 최초로 발생하는 진입주문부터 주문을 발생하기 위한 용도이며

두번째는 주문응답이벤트나 체결통보이벤트 발생 시 최근 주문의 상태를 확인하는 용도입니다.

전역변수로 선언하고 사용합니다.

- ② 현재가로 매도주문을 발생하고 주문식별번호는 SID변수에 저장합니다.

※ OrderCode를 주문종목으로 지정

※ 주문수량으로 지정한 Signal.count는 차트시스템에서 설정된 수량입니다.

※ 저장된 주문식별번호는 주문응답 이벤트 발생 시 현재 주문의 응답인지 확인하기 위해 사용됩니다.

- ③ 매도진입이라는 텍스트를 디버깅창과 [예스스팟모니터]에 출력합니다.

- ④ SellFill에 0을 할당합니다.

※ 변수(SellFill)은 Sell신호시 발생한 매도주문이 체결되면 1로 변경됩니다.

- F. ExitShort 완성신호 발생했을 경우로 완성신호 이벤트 발생시 기술하는 내용입니다.

```
if (Position == -1 && Signal.signalKind == 4)
{
    ① if (SellFill == 1)
    {
        ② A1.OrderBuy(OrdeCode, Vol, F1.Ask(2), 0);
        ③ Main.MessageLog("매도청산");
    }
    ④ if (SellFill == 0)
    {
        ⑤ A1.OrderCancel(SNum);
        ⑥ Main.MessageLog("매도청산취소");
    }
}
```

if (Position== -1 and Signal.signalKind == 4) 조건은

완성신호의 종류가 ExitLong이고 Position에 1이 이미 저장된 상태라는 내용입니다.

스팟전략 적용하고 Buy나 Sell신호가 발생해서 Position가 1값을 가진 이후에 발생하게 한 내용입니다.

ExitShort신호가 발생할 때는 진입주문이 미체결된 경우와 체결된 경우로 나누어 조건문을 만듭니다.

①. SellFill은 매도진입주문 시에 0으로 셋팅되고 체결통보를 받으면 1로 변경이 되게 작성되어 있습니다.

그러므로 SellFill이 1이라는 의미는 매도진입주문이 체결된 상태를 나타냅니다.

②. 매도진입주문이 체결된 상태이므로 매도청산을 위해 매수주문을 실행합니다.

※ OrderCode를 주문종목으로 지정

※ 주문수량으로 지정한 Signal.count는 차트시스템에서 설정된 수량입니다.

※ 매도2호가로 지정가 매수주문합니다.

③. 매도청산실행이라는 텍스트를 디버깅창과 [예스스팟모니터]에 출력합니다.

④. SellFill이 0이라는 의미는 매도진입주문이 미체결된 상태를 나타냅니다.

⑤. 매도진입주문이 체결되지 않았으므로 매도청산을 위한 매도주문은 생략하고 미체결된 매도진입주문에 대해 취소주문을 실행합니다.

※ 주문번호는 주문응답 이벤트 발생 대 저장한 Snum입니다.

⑥. 매도청산취소라는 텍스트를 디버깅창과 [예스스팟모니터]에 출력합니다.

#### G. 주문응답 이벤트 발생

```
function Main_OnOrderResponse(OrderResponse)
{
    ① if (Position == 1 && OrderResponse.orderID == BID)
    {
        ② BNum = OrderResponse.orderNum;
    }
    ③ if (Position == -1 && OrderResponse.orderID == SID)
    {
        ④ SNum = OrderResponse.orderNum;
    }
}
```

Main객체의 OnOrderResponse 이벤트는 주문응답이 발생하면 호출되는 이벤트입니다.

OnOrderResponse 이벤트는 스팟전략에서 발생된 주문에 대한 응답이 발생했을 때만 호출됩니다.

다른 주문화면에서 발생된 주문에 대한 응답에 대해서는 호출되지 않습니다.

발생한 주문응답에 대한 정보는 주문응답객체(OrderResponse)로 확인합니다.

① 최근 buy신호가 발생했고 주문응답의 주문식별번호가 매수진입주문 시 저장한 주문식별번호와 같으면

② 주문번호를 Bnum변수에 저장합니다.

③ 최근 Sell신호가 발생했고 주문응답의 주문식별번호가 매도진입주문 시 저장한 주문식별번호와 같으면

④ 주문번호를 Snum변수에 저장합니다.

※ 완성신호 이벤트 안에 Buy신호 발생하면 Position은 1이고 Sell신호 발생하면 Position은 -1을 가지도록 작성되어 있습니다.

#### H. 체결통보 이벤트 발생

Main객체의 OnNotifyFill이벤트는 체결통보가 발생하면 호출되는 이벤트입니다.

OnNotifyFill이벤트는 전체 주문화면에서 발생한 모든 주문에 대한 체결통보에 대해 호출됩니다.

발생한 체결통보에 대한 정보는 체결통보객체(NotifyFill)로 확인합니다.

주문응답객체가 제공하는 정보는 아래와 같습니다.

NotifyFill	객체	주문체결통보 객체
accountNum	문자열	계좌번호
accountKind	정수	계좌종류
code	문자열	종목코드(단축코드)
fillCount	정수	체결수량
fillPrice	실수	체결가격
fillType	정수	체결유형(1 : 주문, 2 : 정정, 3 : 취소, 11 : 체결, 12 : 정정확인, 13 : 취소...)
error	문자열	에러메세지
orderCondition	정수	주문조건(IOC, FOK, ...)
orderCount	정수	주문수량
orderKind	정수	주문구분(1 : 매도, 2 : 매수)
orderNum	문자열	주문번호
orderPrice	실수	주문가격
orderType	정수	주문유형(1 : 현금매도, 2 : 현금매수, 3 : 신용매도, 4 : 신용매수, ...)
orgOrderNum	문자열	원주문번호
priceKind	정수	가격구분(0 : 지정가, 1 : 시장가, 2 : 최유리지정가, 3 : 지정가(IOC), 4 : ...)

```
function Main_OnNotifyFill(NotifyFill)
{
    ① if (Position == 1 && NotifyFill.orderNum == BNum)
    {
        ② BuyFill = 1;
    }
    ③ if (Position == -1 && NotifyFill.orderNum == SNum)
    {
        ④ SellFill = 1;
    }
}
```

① 최근 buy신호가 발생했고 주문번호가 주문응답이벤트 발생할 때 저장한 주문번호와 같으면

I. ② BuyFill에 1을 저장합니다.

③ 최근 Sell신호가 발생했고 주문번호가 주문응답이벤트 발생할 때 저장한 주문번호와 같으면

④ SellFill에 1을 저장합니다.

#### 4) 스팟수식

/\*스크립트시작-----\*/

var Position;

var OrderCode;

```

var BID;
var SID;
var Snum;
var Bnum;
var BuyFill;
var SellFill;

function Main_OnStart()
{
    Main.MessageLog("시작");

    Position = 0;
    OrdeCode = Main.GetOrderCode(F1.code);
}

function C1_OnRiseSignal(Signal)
{
    Main.MessageLog("신호완성/" + Signal.signalKind);

    if (Signal.signalKind == 1 )
    {
        Position = 1;
        BID = A1.OrderBuy(OrdeCode, Signal.count,F1.current, 0);
        Main.MessageLog("매수진입");
        BuyFill = 0;
    }

    if (Position == 1 && Signal.signalKind == 2 )
    {
        if (BuyFill == 1)
        {
            A1.OrderSell(OrdeCode, Vol,F1.Bid(2), 0);
            Main.MessageLog("매수청산실행");
        }
        if (BuyFill == 0)
        {
            A1.OrderCancel(BNum);
        }
    }
}

```

```

        Main.MessageLog("매수청산취소");
    }

}

if (Signal.signalKind == 3 )
{
    Position = -1;
    SellFill = 0;
    SID = A1.OrderSell(OrdeCode, Signal.count,F1.current, 0);
    Main.MessageLog("매도진입");
}

if (Position == -1 && Signal.signalKind == 4 )
{
    if (SellFill == 1)
    {
        A1.OrderBuy(OrdeCode, Vol,F1.Ask(2), 0);
        Main.MessageLog("매도청산");
    }
    if (SellFill == 0)
    {
        A1.OrderCancel(SNum);
        Main.MessageLog("매도청산취소");
    }
}

```

```

function Main_OnOrderResponse(OrderResponse)
{
    if (Position == 1 && OrderResponse.orderID == BID)
    {
        BNum = OrderResponse.orderNum;
    }
    if (Position == -1 && OrderResponse.orderID == SID)
    {

```

```

SNum = OrderResponse.orderNum;
}

}

function Main_OnNotifyFill(NotifyFill)
{
    if (Position == 1 && NotifyFill.orderNum == BNum)
    {
        BuyFill = 1;
    }
    if (Position == -1 && NotifyFill.orderNum == SNum)
    {
        SellFill = 1;
    }
}
/*스크립트 끝-----*/

```

#### 예제5. 차트연동주문

차트에서 신호발생시 스팟으로 주문을 내는 수식입니다.

차트에서 진입신호가 발생하면 현재가로 주문 후 30초 후에 미체결이 있으면 취소를 하고 청산신호가 발생하면 미체결된 수량을 제외하고 상대5호가로 주문을 집행하게 됩니다.

##### 스크립트 객체화면 설정

차트객체 : 객체명 Chart1 → 속성에서 차트와 동일 아이디 지정

계좌객체 : 객체명 Account1 → 속성에서 계좌번호 선택

```

var OC, MK, pst;
var BID, SID, BNUM, SNUM;
var 취소주문시간 = 30;//30초

```

```

function Main_OnStart()
{
    Main.MessageList("Start");
}

```

//차트종목 종목코드

```

OC = Main.GetOrderCode(Chart1.GetCode(1));
//종목객체 요청
Main.ReqMarketData(OC);
pst = 0;

}

function Main_OnRcvMarketData(MarketData)
{
    if (MarketData.code == OC)
    {
        MK = MarketData;
    }
}

function Chart1_OnRiseSignal(Signal)
{
    //매수진입신호 발생
    if (Signal.signalKind == 1)
    {

        //신호수량 저장
        BV = Signal.count;
        //매수주문 집행(신호수량, 현재가 지정가)
        BID = Account1.OrderBuy(OC, BV, MK.current,0);
        //포지션은 1
        pst = 1;
        //1번 300초 타이머 셋팅
        Main.SetTimer(1, 취소주문시간*1000);
        Main.MessageList("매수진입신호:",BV);
    }

    //스팟에서 매수진입 이후에 매수청산 신호 발생
    if (pst == 1 && Signal.signalKind == 2)
    {
        //1번 타이머 종료
    }
}

```

```

Main.KillTimer(1);

//진입매수주문 미체결 객체 셋팅
Account1.SetUnfillOrderNumber(BNUM);
//미체결이 있으면
if ( Account1.Unfill.count > 0 )
{
    //진입수량에서 미체결수량 차감
    BV = BV - Account1.Unfill.count;
    //미체결 주문 취소
    Account1.OrderCancel(BNUM);
}

Main.MessageList("매수청산:",BV);
//매수진입수량에서 미체결수량 차감후 0이 아니면(진입전량 체결이나 일부체결)
if (BV > 0)
{
    //청산을 위해 매도주문 집행(수량은 BV, 매수5호가 지정가)
    Account1.OrderSell(OC, BV, MK.Bid(5), 0);
}

//매도진입신호 발생
if (Signal.signalKind == 3)
{
    //신호수량 저장
    SV = Signal.count;
    //매도주문 집행(신호수량, 현재가 지정가)
    SID = Account1.OrderSell(OC, SV, MK.current, 0);
    //포지션은 -1
    pst = -1;
    //3번 300초 타이머 셋팅
    Main.SetTimer(3, 취소주문시간*1000);
    Main.MessageList("매도진입신호:",SV);
}

```

```

//스팟에서 매도진입 이후에 매도청산 신호 발생
if (pst == -1 && Signal.signalKind == 4)
{
    //3번 타이머 종료
    Main.KillTimer(3);

    //진입매도주문 미체결 객체 셋팅
    Account1.SetUnfillOrderNumber(SNUM);
    //미체결이 있으면
    if ( Account1.Unfill.count > 0 )
    {
        //진입수량에서 미체결수량 차감
        SV = SV-Account1.Unfill.count;
        //미체결 주문 취소
        Account1.OrderCancel(SNUM);
    }
    Main.MessageList("매도청산:",SV);
    //매도진입수량에서 미체결수량 차감후 0이 아니면(진입전량 체결이나 일부체결)
    if (SV > 0)
    {
        //청산을 위해 매수주문 집행(수량은 SV, 매도5호가 지정가)
        Account1.OrderBuy(OC, SV, MK.Ask(5), 0);
    }
}
}

//주문응답
function Main_OnOrderResponse(OrderResponse)
{
    //매수진입주문에 대한 응답이 오면
    if (OrderResponse.orderID == BID)
    {
        //주문번호 저장
        BNUM = OrderResponse.orderNum;
    }
}

```

```

//매도진입주문에 대한 응답이 오면
if (OrderResponse.orderID == SID)
{
    //주문번호 저장
    SNUM = OrderResponse.orderNum;
}

//타이머 실행
function Main_OnTimer(nEventID)
{
    //1번타이머
    if (nEventID == 1)
    {
        //타이머 종료
        Main.KillTimer(1);

        //매수진입주문에 대해 미체결 셋팅
        Account1.SetUnfillOrderNumber(BNUM);
        //미체결이 있으면
        if (Account1.Unfill.count > 0)
        {
            //매수진입수량에서 미체결수량 차감
            BV = BV - Account1.Unfill.count;
            Main.MessageList("매수진입취소:",BV);
            //미체결주문 취소
            Account1.OrderCancel(BNUM);
        }
    }

    //3번타이머
    if (nEventID == 3)
    {
        //타이머 종료
        Main.KillTimer(3);
    }
}

```

```

//매도진입주문에 대해 미체결 설정
Account1.SetUnfillOrderNumber(SNUM);
//미체결이 있으면
if (Account1.Unfill.count > 0)
{
    //매수진입수량에서 미체결수량 차감
    SV = SV-Account1.Unfill.count;
    Main.MessageList("매도진입취소:",SV);
    //미체결주문 취소
    Account1.OrderCancel(SNUM);
}
}
}

```

기존 예스랭귀지로 구현되는 전략들 중에 선물의 움직임 참조하여 콜옵션과 풋옵션을 동시에 거래하는 전략들이 많이 있습니다.

#### **예제6. 합성선물**

예스스팟으로 합성선물전략을 구현해 보도록 하겠습니다.

기존 예스랭귀지로 구현되는 전략들 중에 선물의 움직임 참조하여 콜옵션과 풋옵션을 동시에 거래하는 전략들이 많이 있습니다.

예스랭귀지로 이를 구현할 때 어려움 중 하나는 차트에서는 주종목(data1)에 대해서만 주문이 가능하므로 하나의 차트에서는 복수 종목에 대해 주문을 발생할 수 없다는 것입니다. 그러므로 콜옵션이 주종목인 차트와 풋옵션이 주종목인 차트를 각각 만들어 각각 시스템을 적용해야 하는 어려움이 있습니다.

또한 원래 선물전략에서는 Atstop이나 Atlimit타입이나 혹은 강제청산 함수로 주문함수가 작성되어 있어 봉 미완성시에 신호가 발생하는데 선물이 참조데이터가 되어 참조데이터에서 조건을 판단해야 하므로 모두 봉 완성시로 판단하게 변경 작성될 수 밖에 없는 문제도 있습니다.

예스스팟을 이용하면 선물전략은 변경없이 그대로 차트에 적용하고  
차트에서 신호만 받아와 콜옵션과 풋옵션에 동시에 주문을 집행하여 합성전략 구현을 할 수 있습니다.

##### 1) 전략내용

###### A. 선물차트에서 매수신호가 발생하면 콜매수+풋매도

선물차트에서 매수청산신호가 발생하면 매수신호시 주문종목 청산

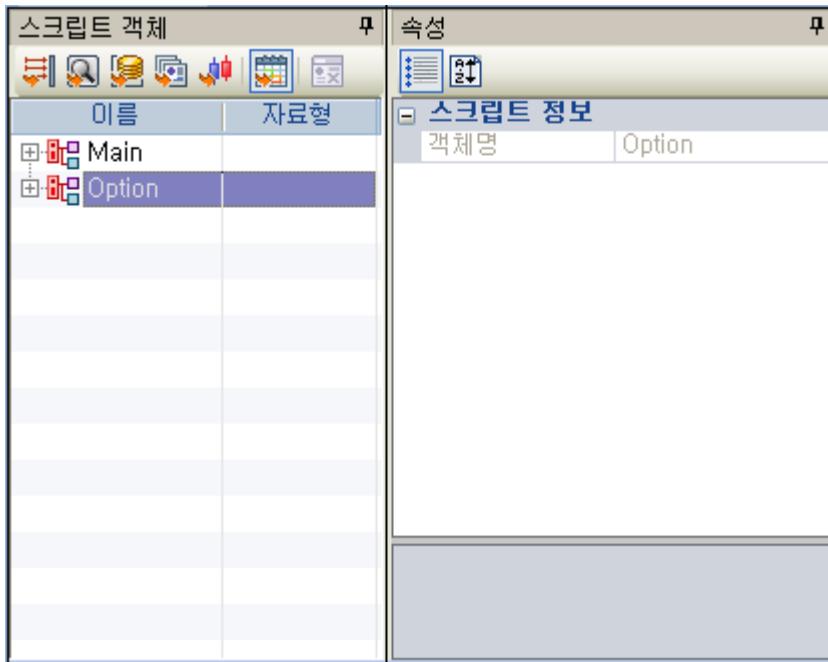
선물차트에서 매도신호가 발생하면 콜매도+풋매수

선물차트에서 매도청산신호가 발생하면 매도신호시 주문종목 청산

2) 스크립트 객체화면 설정

A. 옵션객체

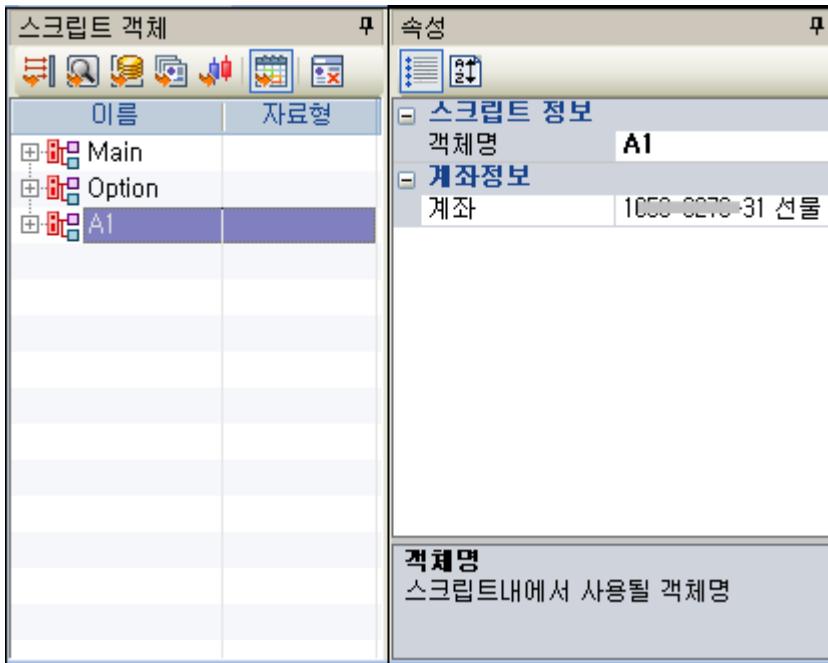
옵션 종목에 주문을 내므로 옵션정보를 알 수 있는 옵션객체가 필요합니다.



B. 계좌객체

주문을 위해 계좌객체가 필요합니다.

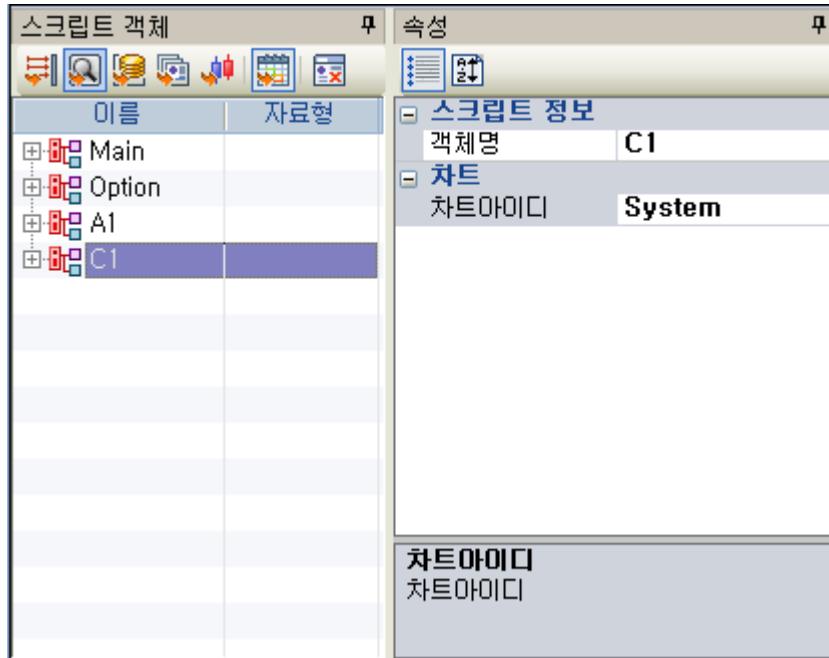
예제에서는 객체명을 A1으로 지정하고 계좌는 선물옵션계좌로 설정합니다.



### C. 차트객체

차트에 적용된 시스템 신호정보를 알아야 하므로 차트객체가 필요합니다.

예제에서는 객체명을 C1으로 하고 차트아이디는 System으로 합니다.



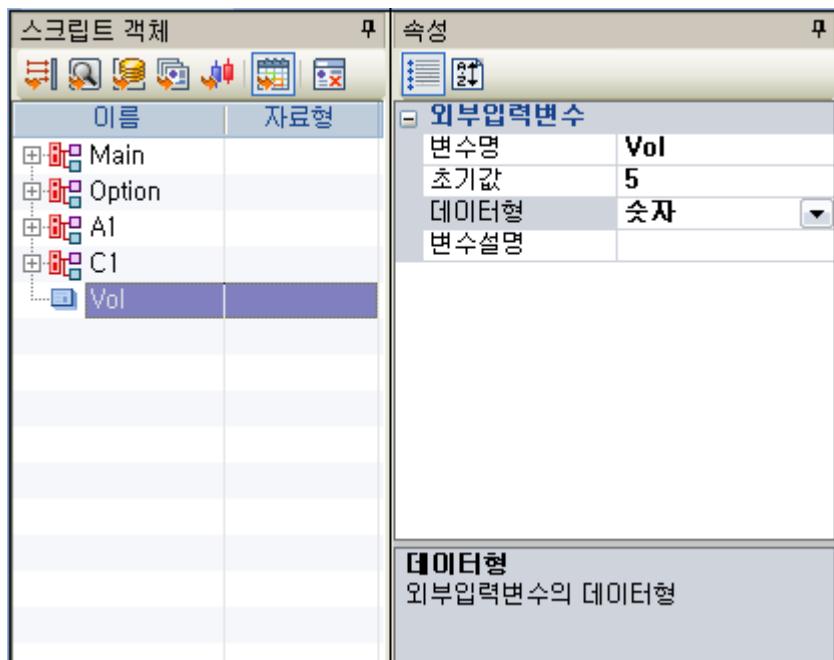
선물시스템이 적용된 차트에 아이디를 System으로 지정하셔야 해야 차트객체가 해당 차트에서 신호가 발생한 것을 알 수 있습니다.



차트에서 직접 주문이 발생하는 것이 아니므로 시스템은 시험적용으로 하시면 됩니다.

### D. 외부변수

주문수량을 [예스스팟 모니터]에서 변경할 수 있게 외부변수를 추가합니다.



### 3) 전략작성

- A. 스팟전략이 최초 실행될 때 [예스스팟 스튜디오]의 디버깅창과 [예스스팟 모니터]에 시작이라는 텍스트를 출력합니다.

```
1 var Start;
2
3 function Main_OnStart()
4 {
5     Main.MessageLog("시작");
6     Start = 0;
7 }
```

또한 스팟전략 실행 후에 차트에서 최초 신호가 청산신호 일 수 있습니다.

스팟전략 실행 후에 차트에서 발생하는 최초의 진입신호부터 주문을 하기 위한 처리가 필요하므로 전역변수(Start)를 만들어 우선 기초값을 0으로 셋팅합니다.

- B. 차트 신호와 연계된 전략이므로 차트객체(C1)의 OnRiseSignal 이벤트를 사용합니다.

```

1  var Start;
2
3  function Main_OnStart()
4  {
5      Main.MessageLog("시작");
6      Start = 0;
7  }
8
9  function C1_OnRiseSignal(Signal)
10 {
11     Main.MessageLog("신호완성/" + Signal.signalKind);
12 }

```

OnRiseSignal은 연결된 차트에서 완성신호가 발생했을 때 호출되는 이벤트이고 완성신호 정보는 완성신호객체(Signal)가 제공합니다.

메시지로그 함수로 완성신호 이벤트가 발생하면 신호완성이라는 텍스트를 함께 신호종류를 디버깅창과 [예스스팟모니터]에 출력합니다.

완성신호의 종류(Signal.signalKind)는 1~4까지 정수로 리턴됩니다.

Buy : 1, Exitlong : 2, Sell : 3, Exitshort : 4

C. 완성신호 이벤트 발생할 때 신호종류만 확인하고 주문을 발생하면 되므로 이하는 모두 완성신호 이벤트 안에 기술합니다.

Buy 완성신호가 발생했을 경우

```

if (Signal.signalKind == 1 )
{
    ① Start = 1;
    ② BuyCallCode = Option.GetATMCallRecent(0);
    ③ BuyPutCode = Option.GetATMPutRecent(0);
    ④ var BuyCallPrice = Option.GetAskByCode(BuyCallCode, 2);
    ⑤ var BuyPutPrice = Option.GetBidByCode(BuyPutCode, 2);
    ⑥ A1.OrderBuy(BuyCallCode, Vol, BuyCallPrice, 0);
    ⑦ A1.OrderSell(BuyPutCode, Vol, BuyPutPrice, 0);
    ⑧ Main.MessageLog("합성선물매수");
}

```

`if (Signal.signalKind == 1)` 조건은 완성신호의 종류가 buy라는 내용입니다.

①. 변수(Start)에 1을 저장합니다.

※ 차트에서 최초로 진입신호가 발생한 후부터 주문을 집행하기 위한 용도입니다.

②. 변수(BuyCallCode)에 현재 시점의 ATM 등가격 콜옵션의 종목코드를 저장합니다

※ Buy신호시 저장된 콜옵션 종목코드는 현재 주문에서 주문종목코드로 이용하고 이후 Exitlong신호 발생할 때 콜옵션을 청산하기 위한 주문종목코드로 이용합니다.

저장된 값이 다음 Exitlong신호시에 사용되어야 하므로 해당 변수들은 전역변수로 선언합니다.

- ③. 변수(BuyPutCode)에 현재 시점의 ATM 등가격 풋옵션의 종목코드를 저장합니다.
- \* Buy신호시 저장된 풋옵션 종목코드는 현재 주문에서 주문종목코드로 이용하고 이후 Exitlong신호 발생할 때 풋옵션을 청산하기 위한 주문종목코드로 이용합니다.
- 저장된 값이 다음 Exitlong신호시에도 사용되어야 하므로 해당 변수들은 전역변수로 선언합니다.
- ④. 변수(BuyCallPrice)에 현재 시점의 ATM 등가격 콜옵션의 매도2호가를 저장합니다.
- \* 콜옵션 매수주문시 주문가격으로 이용하기 위한 용도입니다.
- buy신호 발생할 때만 사용되므로 지역변수로 선언해서 사용합니다.
- ⑤. 변수(BuyPutPrice)에 현재 시점의 ATM 등가격 풋옵션의 매수2호가를 저장합니다.
- \* 풋옵션 매도주문시 주문가격으로 이용하기 위한 용도입니다.
- buy신호 발생할 때만 사용되므로 지역변수로 선언해서 사용합니다.
- ⑥. ATM 등가격 콜옵션을 매도2호가로 5계약 매수주문합니다.
- \* 종목코드는 BuyCallCode, 수량은 외부변수, 지정가로 BuyCallPrice가 주문가격입니다.
- ⑦. ATM 등가격 풋옵션을 매수2호가로 5계약 매도주문합니다.
- \* 종목코드는 BuyPutcode, 수량은 외부변수, 지정가로 BuyPutPrice가 주문가격입니다.
- ⑧. 메시지로그 함수를 이용해 합성선물매수라는 텍스트를 출력합니다.

ExitLong 완성신호가 발생 했을 경우

```
if (Start == 1 && Signal.signalKind == 2 )
{
    ① var BxCallPrice = Option.GetBidByCode(BuyCallCode, 2);
    ② var BxPutPrice = Option.GetAskByCode(BuyPutCode, 2);
    ③ A1.OrderSell(BuyCallCode, Vol, BxCallPrice, 0);
    ④ A1.OrderBuy(BuyPutCode, Vol, BxPutPrice, 0);
    ⑤ Main.MessageLog("합성선물매수청산");
}
```

if (Start == 1 and Signal.signalKind == 2) 조건은 완성신호의 종류가 ExitLong이고 Start에 1이 이미 저장된 상태라는 내용입니다.

스팟전략 적용하고 Buy나 Sell신호가 발생해서 Start가 1값을 가진 이후에 발생하게 한 내용입니다.

- ①. 변수(BxCallPrice)에 Buy신호시 매수 주문한 콜옵션 종목의 매수2호가를 저장합니다.
- ②. 변수(BXPallPrice)에 Buy신호시 매도 주문한 풋옵션 종목의 매도2호가를 저장합니다.
- ③. ATM 등가격 콜옵션을 매수2호가로 5계약 매도주문합니다.
- \* 종목코드는 BuyCallcode, 수량은 외부변수, 주문가격은 지정가로 BxCallPrice가 주문가격입니다.
- ④. ATM 등가격 풋옵션을 매도2호가로 5계약 매수주문합니다.
- \* 종목코드는 BuyPutcode, 수량은 외부변수, 주문가격은 지정가로 BxPutprice가 주문가격입니다.
- ⑤. 메시지로그 함수를 이용해 합성선물매수청산이라는 텍스트를 출력합니다.

Sell 완성신호가 발생했을 경우

```

if (Signal.signalKind == 3 )
{
    ① Start = 1;
    ② SellCallCode = Option.GetATMCallRecent(0);
    ③ SellPutCode = Option.GetATMPutRecent(0);
    ④ var SellCallPrice = Option.GetBidByCode(SellCallCode,2);
    ⑤ var SellPutPrice = Option.GetAskByCode(SellPutCode,2);
    ⑥ A1.OrderSell(SellCallCode, Vol, SellCallPrice , 0);
    ⑦ A1.OrderBuy(SellPutCode, Vol, SellPutPrice, 0);
    ⑧ Main.MessageLog("합성선물매도");
}

```

if (Signal.signalKind == 3) 조건은 완성신호의 종류가 Sell이라는 내용입니다.

①. 변수(Start)에 1을 저장합니다.

※ 차트에서 최초로 진입신호가 발생한 후부터 주문을 집행하기 위한 용도입니다.

②. 변수(SellCallCode)에 현재 시점의 ATM 등가격 콜옵션의 종목코드를 저장합니다

※ Sell신호시 저장된 콜옵션 종목코드는 현재 주문에서 주문종목코드로 이용하고 이후 ExitShort신호 발생할 때 콜옵션을 청산하기 위한 주문종목코드로 이용합니다.

저장된 값이 다음 Exitlong신호시에 사용되어야 하므로 해당 변수들은 전역변수로 선언합니다.

③. 변수(SellPutCode)에 현재 시점의 ATM 등가격 풋옵션의 종목코드를 저장합니다.

※ Sell신호시 저장된 풋옵션 종목코드는 현재 주문에서 주문종목코드로 이용하고 이후 ExitShort신호 발생할 때 풋옵션을 청산하기 위한 주문종목코드로 이용합니다.

저장된 값이 다음 Exitlong신호시에도 사용되어야 하므로 해당 변수들은 전역변수로 선언합니다.

④. 변수(SellCallPrice)에 현재 시점의 ATM 등가격 콜옵션의 매수2호가를 저장합니다.

※ 콜옵션 매도주문시 주문가격으로 이용하기 위한 용도입니다.

Sell신호 발생할 때만 사용되므로 지역변수로 선언해서 사용합니다.

⑤. 변수(SellPutPrice)에 현재 시점의 ATM 등가격 풋옵션의 매도2호가를 저장합니다.

※ 풋옵션 매수주문시 주문가격으로 이용하기 위한 용도입니다.

Sell신호 발생할 때만 사용되므로 지역변수로 선언해서 사용합니다.

⑥. ATM 등가격 콜옵션을 매수2호가로 5계약 매도주문합니다.

※ 종목코드는 SellCallCode, 수량은 외부변수, 지정가로 SellCallPrice가 주문가격입니다.

⑦. ATM 등가격 풋옵션을 매도2호가로 5계약 매수주문합니다.

※ 종목코드는 SellPutCode, 수량은 외부변수, 지정가로 SellCallPrice가 주문가격입니다.

⑧. 메시지로그 함수를 이용해 합성선물매도라는 텍스트를 출력합니다.

ExitShort 완성신호가 발생 했을 경우

```

if (Start == 1 && Signal.signalKind == 4 )
{
    ① var SxCallPrice = Option.GetAskByCode(SellCallCode, 2);
    ② var SxPutPrice = Option.GetBidByCode(SellPutCode, 2);
    ③ A1.OrderBuy(SellCallCode, Vol, SxCallPrice, 0);
    ④ A1.OrderSell(SellPutCode, Vol, SxPutPrice, 0);
    ⑤ Main.MessageLog("합성선물매도청산");
}

```

`if (Start == 1 and Signal.signalKind == 4)` 조건은 완성신호의 종류가 Exitshort이고 Start에 1이 이미 저장된 상태라는 내용입니다.

스팟전략 적용하고 Buy나 Sell신호가 발생해서 Start가 1값을 가진 이후에 발생하게 한 내용입니다.

- ①. 변수(SxCallPrice)에 Sell신호시 매도 주문한 콜옵션 종목의 매도2호가를 저장합니다.
- ②. 변수(SXPutPrice)에 Sell신호시 매수 주문한 풋옵션 종목의 매수2호가를 저장합니다.
- ③. ATM 등가격 콜옵션을 매도2호가로 5계약 매수주문합니다.  
※ 종목코드는 SellCallCode, 수량은 외부변수, 주문가격은 지정가로 SxCallPrice가 주문가격입니다.
- ④. ATM 등가격 풋옵션을 매수2호가로 5계약 매도주문합니다.  
※ 종목코드는 SellPutCode, 수량은 외부변수, 주문가격은 지정가로 SxPutPrice가 주문가격입니다.
- ⑤. 메시지로그 함수를 이용해 합성선물매도청산이라는 텍스트를 출력합니다.

#### 4) 스팟수식

```

/*스크립트시작-----*/
var Start;
var BuyCallCode;
var BuyPutCode;
var SellCallCode;
var SellPutCode;

function Main_OnStart()
{
    Main.MessageLog("시작");
    Start = 0;
}

function C1_OnRiseSignal(Signal)
{
    Main.MessageLog("신호완성/"+Signal.signalKind);
}

```

```

if (Signal.signalKind == 1 )
{
    Start = 1;
    BuyCallCode = Option.GetATMCallRecent(0);
    BuyPutCode = Option.GetATMPutRecent(0);
    var BuyCallPrice = Option.GetAskByCode(BuyCallCode,2);
    var BuyPutPrice = Option.GetBidByCode(BuyPutCode,2);
    A1.OrderBuy(BuyCallCode, Vol, BuyCallPrice, 0);
    A1.OrderSell(BuyPutCode, Vol, BuyPutPrice, 0);
    Main.MessageLog("합성선을매수");
}

if (Start == 1 && Signal.signalKind == 2 )
{
    var BxCallPrice = Option.GetBidByCode(BuyCallCode, 2);
    var BxPutPrice = Option.GetAskByCode(BuyPutCode, 2);
    A1.OrderSell(BuyCallCode, Vol, BxCallPrice, 0);
    A1.OrderBuy(BuyPutCode, Vol, BxPutPrice, 0);
    Main.MessageLog("합성선을매수청산");
}

if (Signal.signalKind == 3 )
{
    Start = 1;
    SellCallCode = Option.GetATMCallRecent(0);
    SellPutCode = Option.GetATMPutRecent(0);
    var SellCallPrice = Option.GetBidByCode(SellCallCode,2);
    var SellPutPrice = Option.GetAskByCode(SellPutCode,2);
    A1.OrderSell(SellCallCode, Vol, SellCallPrice , 0);
    A1.OrderBuy(SellPutCode, Vol, SellPutPrice, 0);
    Main.MessageLog("합성선을매도");
}

if (Start == 1 && Signal.signalKind == 4 )
{
    var SxCallPrice = Option.GetAskByCode(SellCallCode, 2);
    var SxPutPrice = Option.GetBidByCode(SellPutCode, 2);
    A1.OrderBuy(SellCallCode, Vol, SxCallPrice, 0);
}

```

```

        A1.OrderSell(SellPutCode, Vol, SxPutPrice, 0);
        Main.MessageLog("합성선물매도청산");
    }
}

/*스크립트끝-----*/

```

#### **예제7. 주기가 다른 여러 차트를 참조해 주문발생하기**

예스랭귀지에서 타주기의 값을 이용하는 시스템을 만들 때 보통 2가지 방법을 사용합니다.

첫번째 방법은 참조데이터로 타주기 데이터를 추가하고 수식에서 이용하는 것이 있고, 두번째 방법은 주종목 데이터로만 타주기의 계산식을 만들어 사용하는 방법이 있습니다.

타주기 참조데이터를 사용하는 경우에는 수식에서 data2(macd(12,26))과 같이 타종목참조 함수를 이용하면 수식 작성은 간단하지만 최근 완성된 봉까지만 사용되므로 현재 만들어지고 있는 미완성봉의 값을 이용하지 못하는 단점이 있고, 주종목 데이터로만 계산식을 만들어 사용하는 경우에는 계산식 작성이 어렵고, 실시간 차트의 데이터가 최대건수가 5000개봉이므로 봉수가 부족하여 구현이 가능하지 못한 경우도 있으며 3분봉에서 10분봉같이 주종목의 주기의 배수가 아닌 타주기는 구현이 되지 않습니다.

예스스팟에서는 차트객체를 통해 차트에 적용된 지표나 검색, 강조의 값을 실시간으로 최종값 및 이전 봉의 값도 이용할 수 있으므로 타주기의 지표나 계산값을 이용할 때 예스랭귀지 시스템과 연동해 사용하시면 좀더 간단하고 제약 없이 전략 구현이 가능합니다.

#### **1) 전략내용**

종목은 간단한 전략 구현을 위해 일반종목인 삼성전자를 이용합니다.

매수 : 1분봉 차트에서 종가가 파라볼릭을 상향돌파하면 매수하는데

일간 이동평균 5-20-60 정배열이고 5일 이동평균은 상승 중이고

일간 스토캐스틱 슬로우K는 30 이하인 상태일 때만 매수주문

매도 : 1분봉 차트에서 종가가 파라볼릭을 하향이탈하면 매도

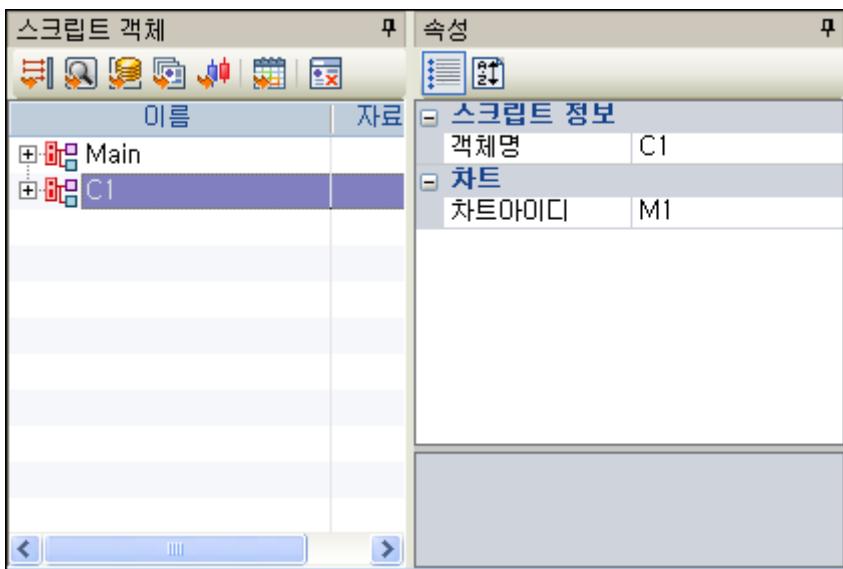
파라볼릭 전략은 확정된 신호를 위해 시스템에서 발생하는 신호를 참고합니다.

#### **2) 스크립트 객체화면 설정**

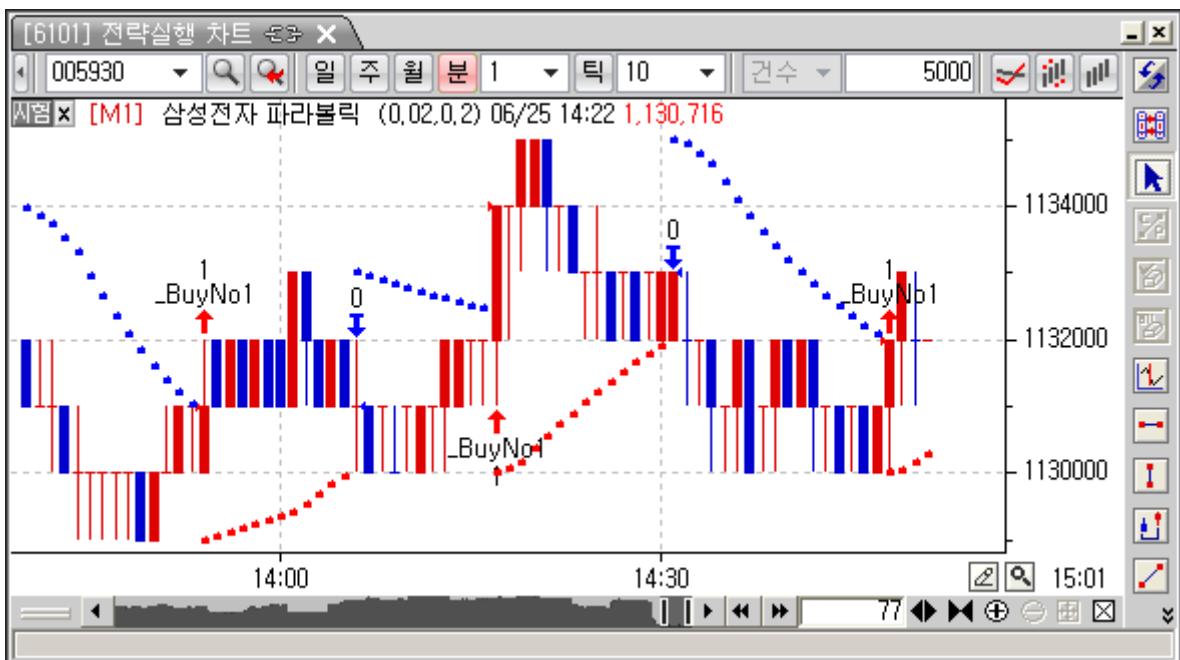
##### **A. 첫번째 차트객체**

첫번째 차트객체는 파라볼릭 신호등 1분봉 차트에 적용된 완성된 신호정보를 가져오는 용도입니다.

차트객체를 추가하고 객체명은 C1으로 하고 차트아이디는 M1으로 합니다.



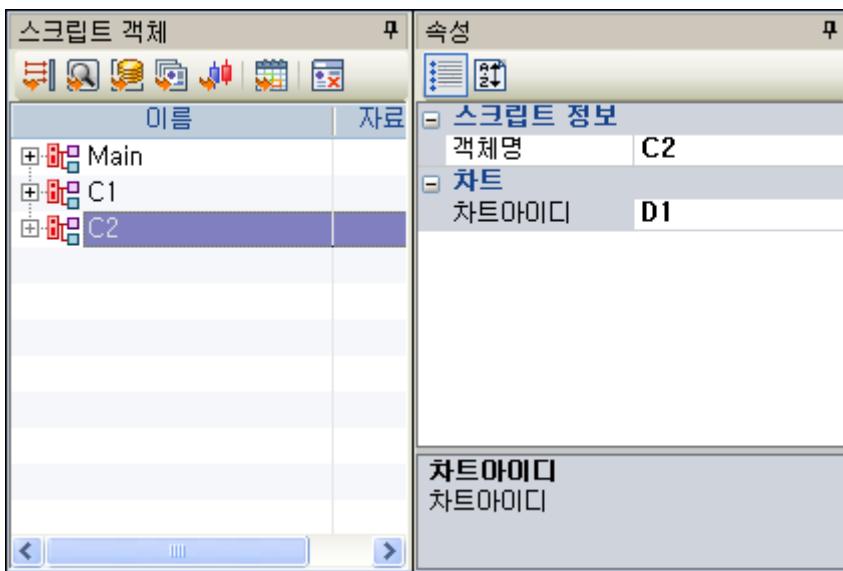
1분봉 차트에 파라볼릭 시스템을 적용하고 차트아이디는 차트객체와 같은 M1으로 지정합니다.



#### B. 두번째 차트객체

두번째 차트객체는 일간의 값을 가져오는 용도입니다.

차트객체를 추가하고 객체명은 C2으로 하고 차트아이디는 D1으로 합니다.



일간차트의 아이디를 D1로 설정하고 전략에서 필요한 지표를 적용해 주시면 됩니다.

예제에서는 이동평균3개와 스토캐스틱 슬로우K값이 필요하므로

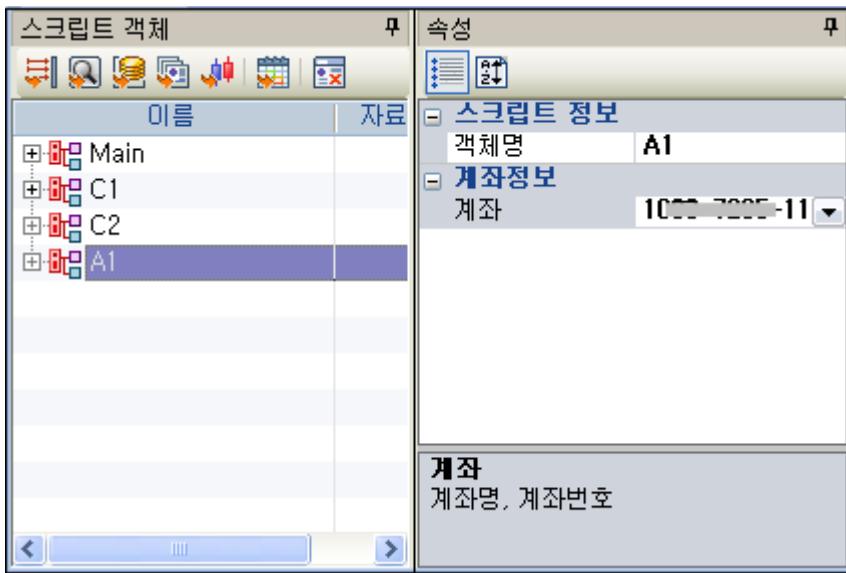
지표식 중 [이동평균 5\_20\_60]과 [Stochastics]을 적용합니다.



### C. 계좌객체

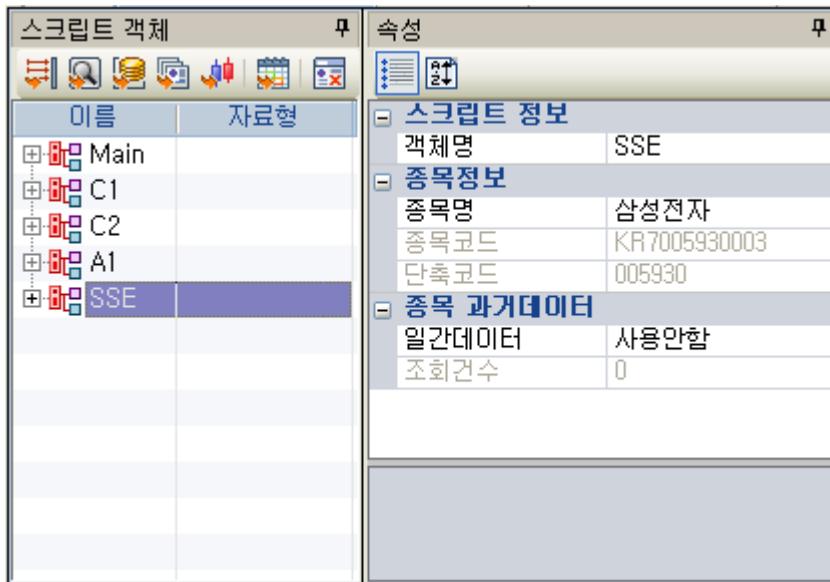
주문을 위해 계좌객체를 추가합니다. 객체명은 A1으로 지정합니다.

계좌는 현물거래 가능계좌로 설정해 주시면 됩니다.



D. 종목객체

주문시 삼성전자의 호가정보를 얻기 위해 종목객체를 추가합니다.



액체명은 SSE로 지정하고 일간데이터는 사용하지 않습니다.

3) 전략작성

A. 스팟전략이 최초 실행될 때.

```

1 var Start;
2
3 function Main_OnStart()
4 {
5     ① Main.MessageLog("스] 작");
6     ② Start = 0;
7 }

```

- ① [예스스팟 스튜디오]의 디버깅창과 [예스스팟 모니터]에 시작이라는 텍스트를 출력합니다.
- ② 스팟전략 실행 후에 차트에서 발생하는 최초의 진입신호부터 주문을 하기 위한 처리가 필요하므로 변수(Start)를 만들어 우선 기초값을 0으로 설정합니다. 전역변수로 선언합니다.

B. 차트 신호와 연계된 전략이므로 차트객체(C1)의 OnRiseSignal 이벤트를 사용합니다.

```

10 function C1_OnRiseSignal(Signal)
11 {
12     ①var dayma1 = C2.GetIndicatorData("이동평균 5_20_60", 1,0);
13     ②var dayma2 = C2.GetIndicatorData("이동평균 5_20_60", 2,0);
14     ③var dayma3 = C2.GetIndicatorData("이동평균 5_20_60", 3,0);
15     ④var predayma1 = C2.GetIndicatorData("이동평균 5_20_60", 1,1);
16     ⑤var slowK = C2.GetIndicatorData("Stochastics",1,0);
17 }

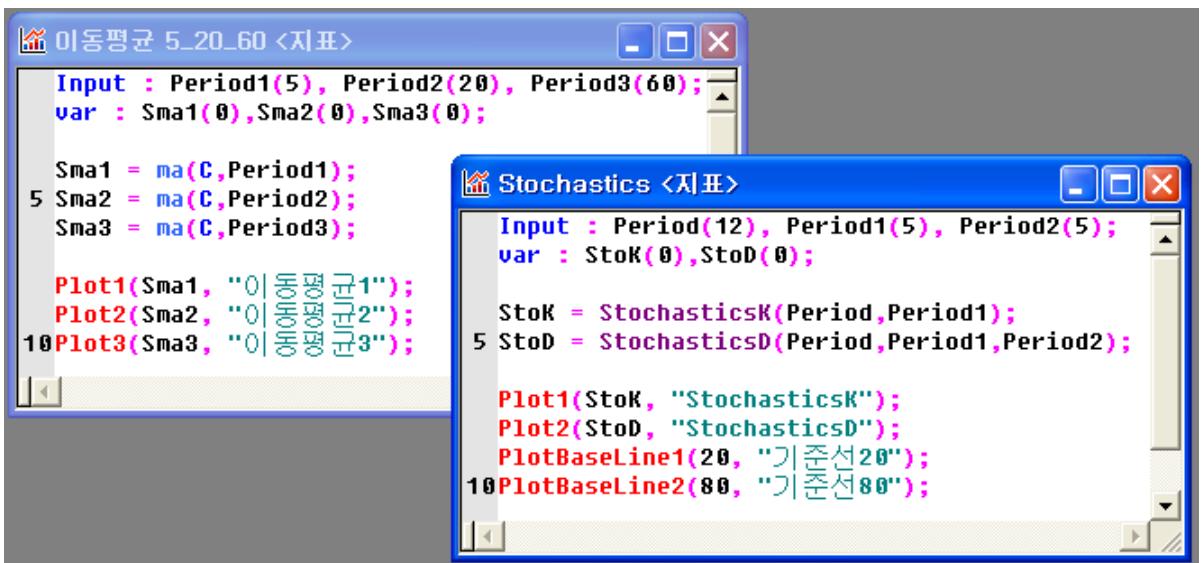
```

OnRiseSignal은 연결된 차트에서 완성신호가 발생했을 때 호출되는 이벤트이고 완성신호 정보는 완성신호객체(Signal)가 제공합니다.

차트(차트객체 C1, 차트아이디 M1)에서 완성신호 이벤트가 발생하면

각 변수에 일간차트에서 필요한 값을 가져와 저장합니다.

- ① 변수(dayma1)에 차트(차트객체 C2, 차트아이디 D1)에 적용된 [이동평균 5\_20\_60]지표의 plot1지표의 현재 값을 가져와 저장합니다.
- ② 변수(dayma2)에 차트(차트객체 C2, 차트아이디 D1)에 적용된 [이동평균 5\_20\_60]지표의 plot2지표의 현재 값을 가져와 저장합니다.
- ③ 변수(dayma3)에 차트(차트객체 C2, 차트아이디 D1)에 적용된 [이동평균 5\_20\_60]지표의 plot3지표의 현재 값을 가져와 저장합니다.
- ④ 변수(predayma1)에 차트(차트객체 C2, 차트아이디 D1)에 적용된 [이동평균 5\_20\_60]지표의 plot1지표의 1봉전 값을 가져와 저장합니다.
- ⑤ 변수(slowk)에 차트(차트객체 C2, 차트아이디 D1)에 적용된 [Stochastics]지표의 plot1지표의 현재 값을 가져와 저장합니다.



GetIndicatorData는 차트의 적용된 지표값을 가져오는 함수입니다.

첫번째 매개변수에 차트에 적용된 지표명을 입력합니다.

두번째 매개변수에는 출력할 plot의 번호를 지정하면 됩니다.

세번째 매개변수에는 이전 봉의 값을 참조할 때 사용합니다.

(순서대로 0은 현재, 1은 1봉전, 2는 2봉전 순입니다.)

C. 조건에 필요한 변수값이 모두 할당되었으므로 매수와 매도식을 작성하면 됩니다.

```

9  function C1_OnRiseSignal(Signal)
10 {
11     var dayma1 = C2.GetIndicatorData("○]동평균 5 20 60", 1,0);
12     var dayma2 = C2.GetIndicatorData("○]동평균 5 20 60", 2,0);
13     var dayma3 = C2.GetIndicatorData("○]동평균 5 20 60", 3,0);
14     var predayma1 = C2.GetIndicatorData("○]동평균 5_20_60", 1,1);
15     var slowK = C2.GetIndicatorData("Stochastics",1,0);
16
17     ⑥ if (Signal.signalKind == 1
18         && dayma1 > dayma2 && dayma2 > dayma3
19         && dayma1 > predayma1
20         && slowK <= 30)
21     {
22         ⑩ A1.OrderBuy(Signal.code,Signal.count,SSE.Ask(2),0);
23         Start = 1;
24     }
25     ⑫ if (Signal.signalKind == 2 && Start == 1)
26     {
27         ⑩ A1.OrderSell(Signal.code,Signal.count,SSE.Bid(2),0);
28     }
29 }
30

```

⑥ 완성신호가 Buy신호이고

- ⑦ 일간차트의 현재시점 이동평균값이 5일 > 20일 > 60일
- ⑧ 일간차트의 5일이동평균의 현재값이 1봉전 값보다 큼
- ⑨ 일간차트의 스토캐스틱 슬로우K가 30 이하이면
- ⑩ 매수주문(완성신호의 종목코드, 완성신호의 수량, 매도2호가, 지정가)하고
- ⑪ Start는 1값
  
- ⑫ 완성신호가 Exitlong신호이고 Start가 10이면
- ⑬ 매도주문(완성신호의 종목코드, 완성신호의 수량, 매수2호가, 지정가)

#### 4) 스팟수식

```

/*스크립트시작-----*/
var Start;

function Main_OnStart()
{
    Main.MessageLog("시작");
    Start = 0;
}

function C1_OnRiseSignal(Signal)
{
    var dayma1 = C2.GetIndicatorData("이동평균 5_20_60", 1,0);
    var dayma2 = C2.GetIndicatorData("이동평균 5_20_60", 2,0);
    var dayma3 = C2.GetIndicatorData("이동평균 5_20_60", 3,0);
    var predayma1 = C2.GetIndicatorData("이동평균 5_20_60", 1,1);
    var slowK = C2.GetIndicatorData("Stochastics",1,0);

    if (Signal.signalKind == 1
        && dayma1 > dayma2 && dayma2 > dayma3
        && dayma1 > predayma1
        && slowK <= 30)
    {
        A1.OrderBuy(Signal.code,Signal.count,SSE.Ask(2),0);
        Start = 1;
    }
    if (Signal.signalKind == 2 && Start == 1)

```

```

{
    A1.OrderSell(Signal.code,Signal.count,SSE.Bid(2),0);
}
/*스크립트끝-----*/

```

#### 예제8. 종목객체를 동적으로 생성해서 이용하기

종목의 호가나 시세정보는 종목객체에서만 제공되므로 예제6과 같이 주문할 때 매도2호가나 매수2호가 등으로 가격을 지정하여 주문을 내려면 종목객체가 필요합니다.

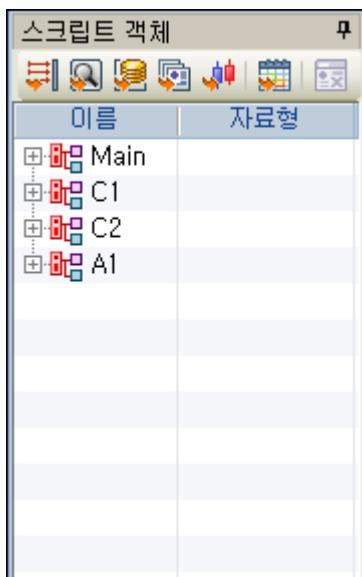
예제6과 같이 전략식에서 단순히 주문가격을 지정할 때만 종목객체가 필요한 경우와 같이 스크립트 객체화면에 종목객체로 설정하지 않고 스크립트 안에서 종목객체를 생성해서 사용할 필요가 있을 때 사용하시면 됩니다.  
예제6을 이용하므로 전략내용과 스크립트 설정은 예제6번과 동일합니다.

#### 1) 전략내용

예제6과 동일합니다.

#### 2) 스크립트 객체설정

예제6과 동일합니다. 스크립트 객체설정 화면에 종목객체는 추가하지 않습니다.



#### 3) 전략작성

##### A. 스팟전략이 최초 실행될 때

	<pre> 6  function Main_OnStart() 7  { 8      ① Main.MessageLog("시작"); 9      ② Start = 0; 10 } </pre>
--	---

- ① [예스스팟 스튜디오]의 디버깅창과 [예스스팟 모니터]에 시작이라는 텍스트를 출력합니다.
- ② 변수(Start)를 만들어 우선 기초값을 0으로 셋팅합니다. 전역변수로 선언합니다.  
스팟전략 실행 후에 차트에서 발생하는 최초의 진입신호부터 주문을 하기 위한 용도입니다.

B. 완성신호 이벤트가 발생하면

```

12     function C1_OnRiseSignal(Signal)
13     {
14         ① Skind = Signal.signalKind;
15         ② Scode = Signal.code;
16         ③ Scount = Signal.count;
17         ④ Main.ReqMarketData(Signal.code, 0);
18     }
19

```

- ① 변수(Skind)에 완성신호의 종류를 저장합니다. 변수는 전역변수로 선언합니다.
- ② 변수(Scode)에 완성신호의 종목코드를 저장합니다. 변수는 전역변수로 선언합니다.
- ③ 변수(Scount)에 완성신호의 주문수량을 저장합니다. 변수는 전역변수로 선언합니다.
- ④ Main객체의 ReqMarketData는 종목객체를 생성을 요청하는 함수입니다.

**Main.ReqMarketData(sItemCode, nDailyCount)**

매개변수로 종목코드와 일간데이터 조회갯수를 지정하면 됩니다

일단데이터의 조회갯수를 0으로 지정하면 일간데이터는 요청하지 않습니다.

C. 종목객체 생성 완료 이벤트가 발생하면

OnRcvMarketData 이벤트는 ReqMarketData함수에 의해 종목객체 요청이 되고 해당 종목객체의 생성이 완료가 되면 호출이 됩니다.

```

20  function Main_OnRcvMarketData(MarketData)
21  {
22      ① var SSEobject = MarketData;
23
24      ② var dayma1 = C2.GetIndicatorData("이동평균 5_20_60", 1,0);
25      var dayma2 = C2.GetIndicatorData("이동평균 5_20_60", 2,0);
26      var dayma3 = C2.GetIndicatorData("이동평균 5_20_60", 3,0);
27      var predayma1 = C2.GetIndicatorData("이동평균 5_20_60", 1,1);
28      var slowK = C2.GetIndicatorData("Stochastics",1,0);
29
30      ③ if (Skind== 1
31          && dayma1 > dayma2 && dayma2 > dayma3
32          && dayma1 > predayma1
33          && slowK <= 30)
34      {
35          ④ A1.OrderBuy(Scode,Scount,SSEobject.Ask(2),0);
36          ⑤ Start = 1;
37
38      }
39      ⑥ if (Start == 1 && Skind == 2)
40      {
41          ⑦ A1.OrderSell(Scode,Scount,SSEobject.Bid(2),0);
42      }
43  }

```

① 변수(SSEobject)에 생성된 종목객체를 저장합니다.

해당 이벤트 내에서만 사용되므로 지역변수로 선언합니다.

② 각 변수에 차트(차트객체 C2, 차트아이디 D1)에 적용된 지정한 이름의 지표의 값을 저장합니다.

해당 이벤트 내에서만 각 변수는 사용되므로 지역변수로 선언합니다.

자세한 설명은 예제6 참고하시기 바랍니다.

③ 저장한 값을 이용해 매수조건문을 만듭니다.

(최근 발생한 신호종류는 매수이고 일간차트의 이동평균은 정배열(5 > 20 > 60)이고

5일 이평은 상승중이고 일간차트의 스토캐스틱 슬로우K가 30이하이면)

④ 매수조건이 만족하면 매수주문을 실행합니다.

완성신호 이벤트가 발생했을 때 변수에 저장한 값을 이용 주문종목코드와 주문수량을 지정합니다.

주문가격은 생성된 종목객체의 매수2호가로 지정합니다.

⑤ 진입이 발생하면 Start에는 1을 저장합니다.

⑥ 매도 조건을 지정합니다.

스팟적용 후 첫 진입주문 이후부터 청산주문 발동하기 위해

Start값은 1이고 완성신호의 종류는 exitlong일때 매도주문합니다.

⑦ 매수와 같이 주문종목코드와 주문수량은 완성신호 이벤트발생할 때 저장한 값을 이용하고

주문가격은 생성된 종목객체의 매수2호가로 지정합니다.

4) 스팟수식

```
/*스크립트시작-----*/  
var Start;  
var Skind;  
var Scode;  
var Scount;  
  
function Main_OnStart()  
{  
    Main.MessageLog("시작");  
    Start = 0;  
}  
  
function C1_OnRiseSignal(Signal)  
{  
    Skind = Signal.signalKind;  
    Scode = Signal.code;  
    Scount = Signal.count;  
    Main.ReqMarketData(Signal.code, 0);  
}  
  
function Main_OnRcvMarketData(MarketData)  
{  
    var SSEobject = MarketData;  
  
    var dayma1 = C2.GetIndicatorData("이동평균 5_20_60", 1,0);  
    var dayma2 = C2.GetIndicatorData("이동평균 5_20_60", 2,0);  
    var dayma3 = C2.GetIndicatorData("이동평균 5_20_60", 3,0);  
    var predayma1 = C2.GetIndicatorData("이동평균 5_20_60", 1,1);  
    var slowK = C2.GetIndicatorData("Stochastics",1,0);  
  
    if (Skind== 1  
        && dayma1 > dayma2 && dayma2 > dayma3  
        && dayma1 > predayma1  
        && slowK <= 30)  
    {
```

```

A1.OrderBuy(Scode,Scount,SSEobject.Ask(2),0);
Start = 1;

}

if (Start == 1 && Skind == 2)
{
    A1.OrderSell(Scode,Scount,SSEobject.Bid(2),0);
}
}

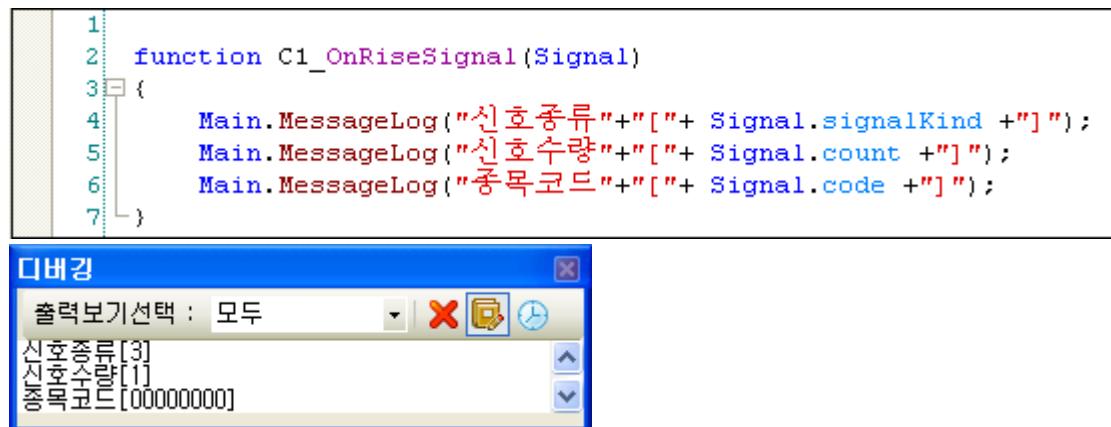
/*스크립트시작-----*/

```

#### 예제9. 사용자정의 함수 만들기

전략에서 자주 사용하는 코드를 사용자정의 함수로 만들면 전략 작성시에 해당 코드를 모두 기술할 필요 없이 만들어진 함수 하나로 대체할 수 있으므로 좀더 간편하게 전략을 작성할 수 있습니다.

아래의 식은 완성신호 이벤트가 발생하면 완성신호의 종류와 수량, 종목코드를 디버깅화면에 출력하게 됩니다.

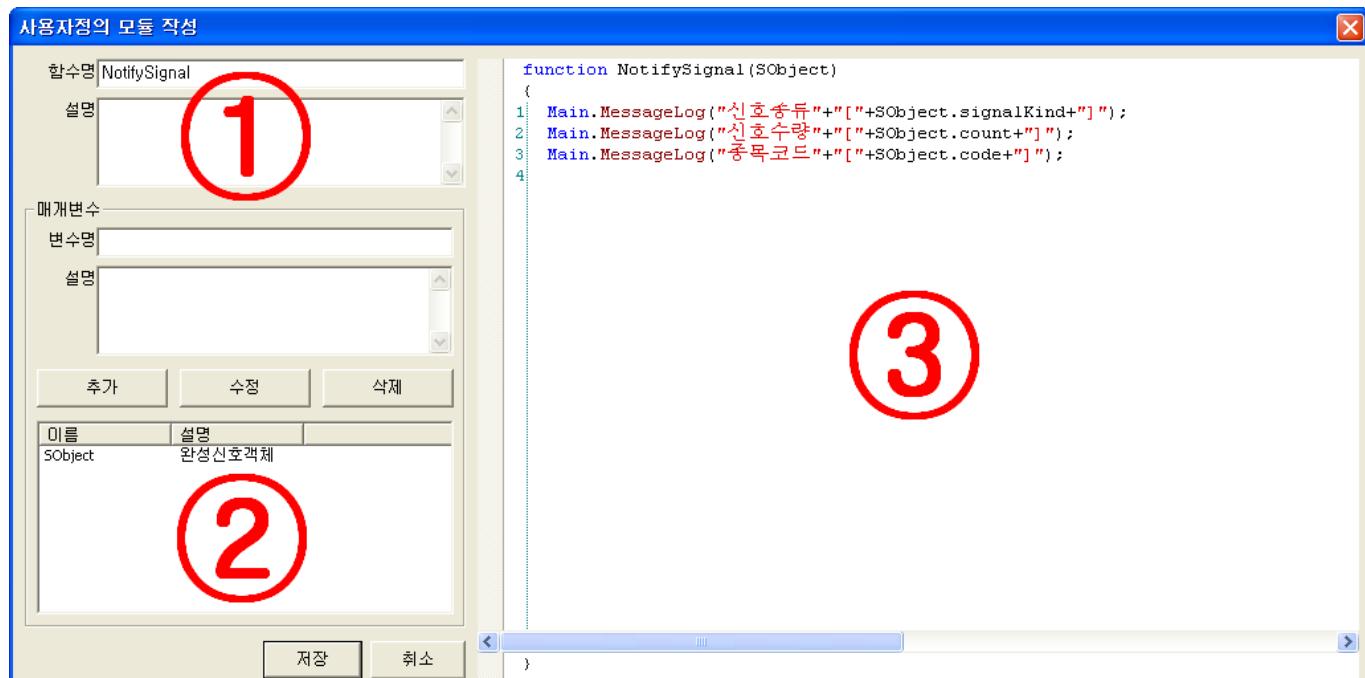


위 내용을 사용자정의 함수로 만들면 출력내용을 모두 기술하지 않고 간단히 함수를 호출해서 처리할 수 있게 됩니다.



1) 사용자정의 모듈 관리자에서 을 클릭하면 사용자정의 함수를 작성할 수 있는 화면이 나타납니다.

사용자정의 모듈 작성 환면에서 함수명과 매개변수를 우선 지정하고 함수식 내용을 작성하시면 됩니다.



- ① 함수명을 입력합니다. 설명은 생략할 수 있습니다.
  - ② 필요한 매개변수를 추가합니다. 매개변수는 복수로 가능하며 설명은 생략할 수 있습니다.
  - ③ 함수식 내용을 입력합니다. 함수식에 함수명과 매개변수는 자동으로 입력되므로 처리내용만 기술하면 됩니다.

```
Main.MessageLog("신호종류"+["+SObject.signalKind+"]");  
Main.MessageLog("신호수량"+["+SObject.count"]);  
Main.MessageLog("종목코드"+["+SObject.code"]);
```

2) 사용자정의 함수식을 작성 후 저장하면 사용자정의 모듈화면에 추가됩니다.

사용자정의 모듈	
이름	설명
<input type="checkbox"/>  할수객체	
<input checked="" type="checkbox"/>  할수	
	<input checked="" type="checkbox"/>  NotifySignal

- 3) 전략에서 사용하고자 하는 사용자정의 할수가 있으면 체크를 하고 전략식에서 사용하면 됩니다.

사용자정의 함수를 더블클릭하면 스크립트 편집창의 커서위치에 함수가 자동 삽입됩니다.

```
1 function C1_OnRiseSignal(Signal)
2 {
3     Main.MessageLog("신호종류"+["+ Signal.signalKind +"] );
4     Main.MessageLog("신호수량"+["+ Signal.count +"] );
5     Main.MessageLog("종목코드"+["+ Signal.code +"] );
6 }
7 
```

```
1 function C1_OnRiseSignal(Signal)
2 {
3     NotifySignal(Signal);
4 }
5 
```

#### 예제10. 사용자정의 함수객체 만들기

특정 객체의 메소드나 프로퍼티를 저장하고 스크립트 내에 다른 이벤트나 함수에서 사용해야 한다면 이를 임시로 저장해야 하는데 저장할 메소드나 프로퍼티의 수 만큼 변수를 선언해 저장하고 사용해야 하므로 코드가 길어지게 됩니다. 사용자정의 함수객체를 만들어 사용하면 코드의 길이도 줄일 수 있고 다른 전략에서도 동일 코드를 반복적으로 작성하는 불편 없이 재사용할 수 있습니다.

- 1) 간단한 예로 아래 식은 S1, S2, S3를 전역변수로 선언하고 완성신호 이벤트가 발생하면 각각 완성신호의 신호종류, 종목코드, 신호수량을 저장해서 스크립트 내에서 사용한 식입니다.

```

1 var S1;
2 var S2;
3 var S3;
4
5 function C1_OnRiseSignal(Signal)
6 {
7     S1 = Signal.signalKind;
8     S2 = Signal.code;
9     S3 = Signal.count;
10    if(Signal.signalKind == 1)
11        A1.OrderBuy(S2, S3, 0, 1);
12}
13
14 function Main_OnOrderResponse(OrderResponse)
15 {
16    if (S1 == 1 &&
17        OrderResponse.code == S2 &&
18        OrderResponse.orderCount == S3)
19    {
20        Main.MessageLog(OrderResponse.orderNum);
21    }
22}

```

변수를 여러 개 사용하지 않고 함수객체를 이용해 작성해 보겠습니다.

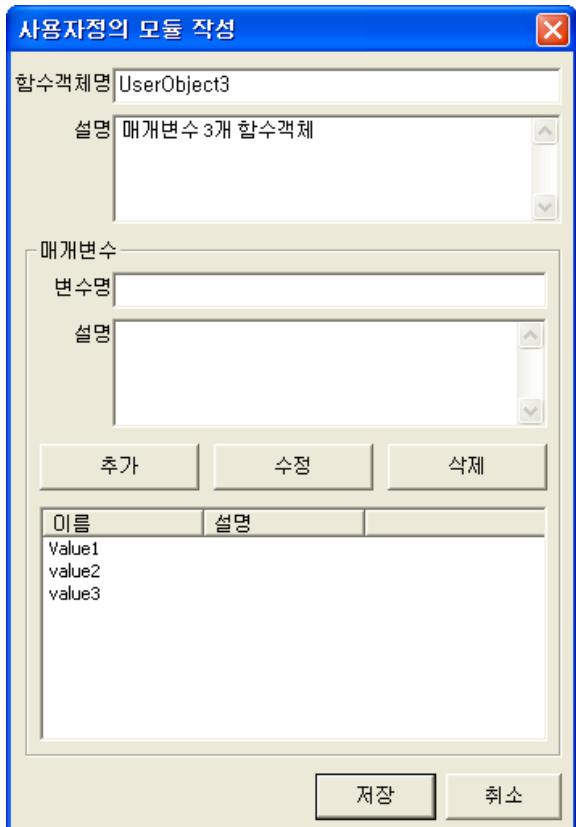
2) 사용자정의 모듈 관리자에서 을 클릭하면 사용자정의 함수객체를 만들 수 있는 화면이 나타납니다.



2) 사용자정의 모듈 작성화면에서 함수객체명을 지정한 후에 원하는 만큼 매개변수를 만들어 추가하면 됩니다.

예제에서 함수객체명은 UserObject3으로 지정하고 저장할 값이 3개이므로 매개변수는 3개를 만듭니다.

매개변수의 이름은 value1, value2, value3으로 지정합니다.



3) 위와 같이 작성하고 저장하면 UserObject3은 매개변수를 3개 가지는 함수객체가 만들어지고 사용자정의 모듈화면에 추가가 됩니다.

사용자정의 모듈	
이름	설명
<input checked="" type="checkbox"/> 함수객체	
<input checked="" type="checkbox"/> UserObject3	매개변수 3개 함수객체
<input type="checkbox"/> 함수	

\* 사용자정의 모듈에서 스크립트에서 사용할 함수객체는 체크 후 사용해야 하며 함수객체명을 더블클릭하면 자동으로 스크립트 편집창의 커서위치에 삽입됩니다.

```

1
2
3  function C1_OnRiseSignal(Signal)
4  {
5      new UserObject(Signal.signalKind, Signal.code, Signal.count);
6

```

5) 함수객체 UserObject3를 이용해 S1,S2,S3를 대신하면 아래와 같습니다.

```

1  var S;
2
3  function C1_OnRiseSignal(Signal)
4  {
5      S = new UserObject(Signal.signalKind, Signal.code, Signal.count);
6      if(S.value1 == 1)
7          A1.OrderBuy(S.value2, S.value3, 0, 1);
8  }
9
10 function Main_OnOrderResponse(OrderResponse)
11 {
12     if (S.value1 == 1 &&
13         OrderResponse.code == S.value2 &&
14         OrderResponse.orderCount == S.value3)
15     {
16         Main.MessageLog(OrderResponse.orderNum);
17     }
18 }

```

기존에 S1,S2,S3에 저장한 값을 함수의 매개변수로 입력하고 함수객체를 전역변수(S)에 할당하면 스크립트에서 최근 완성신호의 종류는 S.value1, 종목코드는 S.value2, 수량은 S.value3로 그 값을 사용할 수 있고 다른 전략에서도 함수객체를 재사용할 수 있습니다.

### 예제11. 동시호가청산

아래 예제는 지정한 시간 이후에 지정한 계좌의 잔고를 모두 청산하는 식입니다.

시간을 판단하기 위해 자바스크립트에서 기본으로 제공하는 Date객체를 이용합니다.

#### 1) 전략내용

15시 6분에 잔고의 모든 종목 청산

#### 2) 스크립트 객체화면 설정

##### A. 계좌객체

계좌객체를 추가한 후 청산 하고자 하는 계좌번호로 지정합니다.

예제에서는 계좌객체명을 A1로 지정하고 선물옵션계좌로 설정합니다.

#### 3) 전략작성

##### A. 스팟전략이 최초 실행될 때

```

1
2     function Main_OnStart()
3     {
4         ① Main.MessageLog("시작")
5         ② Main.SetTimer(1, 5000);
6     }
7

```

- ① [예스스팟 스튜디오]의 디버깅창과 [예스스팟 모니터]에 시작이라는 텍스트를 출력합니다.  
 ② 타이머 함수를 셋팅합니다. 타이머 아이디는 1이고 5초단위로 타이머 이벤트가 발생합니다.

#### B. 타이머 이벤트 발생

```

8     function Main_OnTimer(nEventID)
9     {
10        ① var d = new Date();
11        ② var HHMMDD = d.getHours()*10000+d.getMinutes()*100+d.getSeconds();
12        ③ if (nEventID == 1 && HHMMDD >= 150600)
13        {
14            ④ Main.KillTimer(1);
15            num = A1.GetTheNumberOfBalances();
16            ⑥ for(var i = 0; i < num; i++)
17            {
18                ⑦ A1.SetBalanceIndex(i);
19                ⑧ if (A1.Balance.position == 1)
20                    ⑨ A1.OrderBuy(A1.Balance.code,A1.Balance.count,0,1);
21                ⑩ if (A1.Balance.position == 2)
22                    ⑪ A1.OrderSell(A1.Balance.code,A1.Balance.count,0,1);
23            }
24        }
25    }
26

```

- ① 타이머 이벤트가 발생하면 먼저 Date객체를 변수 d에 할당하여 사용준비를 합니다.  
 ② 시간을 판단해야 하므로 Date객체에서 시/분/초를 가져옵니다.  
 Date객체에서 시간을 반환하는 함수는 getHours()이며 분은 getMinutes(), 초는 getSeconds()입니다.  
 현재시간이 14시30분20초이면 getHours()는 14, getMinutes()는 30, getSeconds()는 20을 반환합니다.  
 3가지 함수를 이용하여 시간을 일반적으로 많이 사용하는 HHMMDD형태로 만들기 위해 아래와 같이 계산식을 만들어 사용합니다.  

$$\text{getHours}() * 10000 + \text{getMinutes}() * 100 + \text{getSeconds}()$$
- ③ 현재 발생한 타이머의 아이디가 지정한 1번이고 시간이 15시06분이후 이면  
 ④ 지정한 시간이 경과했으므로 타이머를 종지합니다.  
 ⑤ for문을 실행하기 위해 A1계좌의 계좌의 종목수를 num에 저장합니다.  
 ⑥ 해당 계좌의 종목 순번이 0부터 시작하므로 0부터 num미만까지 1씩 증가하면서 변수 i에 할당합니다.  
 ⑦ i에 값이 할당될 때마다 i로 지정한 순번에 해당되는 종목으로 잔고객체를 셋팅합니다.  
 ⑧ 셋팅된 잔고객체의 포지션이 매도이면

⑨ 해당 잔고객체의 종목코드와 잔고수량으로 지정하여 시장가 매수주문을 냅니다.

⑩ 셋팅된 잔고객체의 포지션이 매수이면

⑪ 해당 잔고객체의 종목코드와 잔고수량으로 지정하여 시장가 매도주문을 냅니다.

#### 4) 스팟수식

```
/*스크립트시작-----*/
```

```
function Main_OnStart()
{
    Main.MessageLog("시작")
    Main.SetTimer(1, 5000);
}
```

```
function Main_OnTimer(nEventID)
{
    var d = new Date();
    var HHMMDD = d.getHours()*10000+d.getMinutes()*100+d.getSeconds();
    if (nEventID == 1 && HHMMDD >= 150600)
    {
        Main.KillTimer(1);
        num = A1.GetTheNumberOfBalances();
        for(var i = 0; i < num; i++)
        {
            A1.SetBalanceIndex(i);
            if (A1.Balance.position == 1)
                A1.OrderBuy(A1.Balance.code,A1.Balance.count,0,1);
            if (A1.Balance.position == 2)
                A1.OrderSell(A1.Balance.code,A1.Balance.count,0,1);
        }
    }
}
/*스크립트끝-----*/
```

#### 예제12. 특정 가격 이하의 콜/풋 옵션 중 가장 높은 가격의 종목을 선정

아래 예제는 1.0 이하의 콜과 풋 종목 중에 가장 높은 가격의 콜과 풋 종목을 찾아 주문하는 식입니다.

### 1) 전략내용

차트에서 매수신호가 발생하면 콜과 풋의 현재가를 기준으로 1.0이하의 종목을 찾아 매수합니다.

### 2) 스크립트 객체화면 설정

차트객체 : 객체명 Chart1 → 속성에서 차트와 동일 아이디 지정

옵션객체 : 객체명 Option → 속성에서 주기지수옵션을 지정

계좌객체 : 객체명 Account1 → 속성에서 계좌번호 선택

### 3) 스팟수식

```
/*스크립트시작-----*/  
var UNum; var LNum;  
var CallCode; var CallPrice;  
var PutCode; var PutPrice;  
var CC; var PP;  
var CallOrderCode; var PutOrderCode;  
  
//차트에서 신호가 발생하면  
function Chart1_OnRiseSignal(Signal)  
{  
    //해당 신호가 매수신호이면  
    if (Signal.signalKind == 1)  
    {  
        Main.MessageList("-----");  
        Main.MessageList("매수신호 발생");  
        //1.0이하 중 가장 큰 가격을 가지는 종목을 찾음  
        //ATM위 행사가 갯수  
        UNum = Option.uppersATM;  
        //ATM아래 행사가 갯수  
        LNum = Option.lowersATM;  
        //각 행사가의 콜종목의 종목코드를 저장할 변수를 배열변수로 선언  
        CallCode = new Array(UNum+LNum+1);  
        //각 행사가의 콜종목의 현재가를 저장할 변수를 배열변수로 선언  
        CallPrice = new Array(UNum+LNum+1);  
        //각 행사가의 풋종목의 종목코드를 저장할 변수를 배열변수로 선언
```

```

PutCode = new Array(UNum+LNum+1);
//각 행사가의 풋종목의 현재가를 저장할 변수를 배열변수로 선언
PutPrice = new Array(UNum+LNum+1);

//콜종목 찾기
//콜옵션은 ATM기준 위행사가 +단계, 아래가 -단계이므로
//for문에서 LNum의 역수부터 시작해서 UNum까지 1씩 증가하면서 수행하도록 함
for (var i = -LNum; i <= UNum; i++)
{
    //값이 1.0이하이면
    if (Option.GetCurrent(0, i) <= 1.0)
    {
        //해당종목의 현재가를 배열변수 CallPrice의 방번호 i+LNum에 저장
        CallPrice[i+LNum] = Option.GetCurrent(0, i);
        //해당종목의 종목코드를 배열변수 CallCode의 방번호 i+LNum에 저장
        CallCode[i+LNum] = Option.GetATMCallRecent(i);
        //주의
        //배열변수의 방(공간)번호은 -가 없으므로 최하단 행사가를 0번방부터
        //저장하도록 작성해야 함
    }
    else//1.0보다 크면
    {
        //배열변수 CallPrice의 방번호 i+LNum에 -1 저장
        CallPrice[i+LNum] = -1;
        //배열변수 CallCode의 방번호 i+LNum에 -1 저장
        CallCode[i+LNum] = -1;
    }
}

//풋종목 찾기
//풋옵션은 ATM기준 아래 행사가 +단계, 위가 -단계이므로
//for문에서 HNum의 역수부터 시작해서 LNum까지 1씩 증가하면서 수행하도록 함
for (var ii = -UNum; ii <= LNum; ii++)
{
    //ii단계 풋종목이 1.0이하이면
    if (Option.GetCurrent(1, ii) <= 1.0 )

```

```

    {
        //해당종목의 현재가를 배열변수 PutPrice의 방번호 ii+LNum에 저장
        PutPrice[ii+LNum] = Option.GetCurrent(1, ii);

        //해당종목의 현재가를 배열변수 PutCode의 방번호 ii+LNum에 저장
        PutCode[ii+LNum] = Option.GetATMPutRecent(ii);
    }

    else //1.0보다 크면
    {
        //배열변수 PutPrice의 방번호 ii+LNum에 -1 저장
        PutPrice[ii+LNum] = -1;

        //배열변수 PutCode의 방번호 ii+LNum에 -1 저장
        PutCode[ii+LNum] = -1;
    }

    //배열변수 CallPrice의 각 배열방의 값중 가장 큰값을 찾아 CC에 저장하고
    //CallCode의 동일 방번호의 값을 CallOrderCode에 저장
    CC = -1;
    CallOrderCode = -1;
    for (var iii = -LNum; iii <= UNum; iii++)
    {
        if (CallPrice[iii+LNum] > CC)
        {
            CC = CallPrice[iii+LNum];
            CallOrderCode = CallCode[iii+LNum];
        }
    }

    //배열변수 PutPrice의 각 배열방의 값중 가장 큰값을 찾아 PP에 저장하고
    //PutCode의 동일 방번호의 값을 PutOrderCode에 저장
    PP = -1;
    PutOrderCode = -1;
    for (var iiii = -UNum; iiii <= LNum; iiii++)
    {
        if (PutPrice[iiii+UNum] > PP)
        {
            PP = PutPrice[iiii+UNum];
            PutOrderCode = PutCode[iiii+UNum];
        }
    }
}

```

```

        }
    }

    Main.MessageList("-----");
    Main.MessageList("콜종목코드:", CallOrderCode, "/콜현재가 :", CC);
    Main.MessageList("풋종목코드:", PutOrderCode, "/풋현재가 :", PP);

    //콜옵션종목 매수
    Account1.OrderBuy(CallOrderCode, 1, Option.GetAskByCode(CallOrderCode, 2), 1);
    //콜옵션종목 매수
    Account1.OrderBuy(PutOrderCode, 1, Option.GetAskByCode(PutOrderCode, 2), 1);
}

}

/*스크립트끝-----*/

```



### 예제13. 일정주기로 종목검색을 하고 검색된 종목을 매수

일정주기로 종목검색을 하고 검색된 종목에 대해 최대 20종목까지 매수하는 식입니다

9시부터 15시 사이에 1분 주기로 종목검색을 실행하고 검색된 종목이 있으면 최대 20종목까지 매수를 하는 식입니다.

#### 스크립트객체화면 설정

계좌객체를 추가한 후 속성에서 객체명은 Account1으로 지정하고 계좌번호로 지정합니다.

```

var MKList = [];//종목검색 후 종목객체 요청할 종목의 종목코드를 리스트할 배열변수
var MK = [];//종목객체를 저장할 배열변수
var req; //종목객체 요청 횟수를 저장할 변수
var 최대매수종목수 = 20;//매수할 최대 종목수
var 매수금 = 1000000;//종목당 매수금

//스팟 시작
function Main_OnStart()
{
    //1번 타이머, 60초
    Main.SetTimer(1, 60000);
}

//타이머 동작
function Main_OnTimer(nEventID)
{
    //자바스크립트 날짜시간 객체에서 값을 가져와 HHMMSS형식으로 시간 계산
    var d = new Date();
    HHMMSS = d.getHours()*10000+d.getMinutes()*100+d.getSeconds();

    //1번 타이머 동작하고 9시~15시 사이이면
    if (nEventID == 1 && HHMMSS >= 90000 && HHMMSS <= 150000)
    {
        //타이머 종료
        Main.KillTimer(1);
        //지정한 이름의 사용자조건검색을 실행
        Main.ReqPowerSearch("사용자검색조건명");
    }

    if (nEventID == 2)
    {
        Main.KillTimer(2);
        //재요청
        Main.ReqMarketData(MKList[req]);
    }
}

```

```

}

//종목검색완료
function Main_OnRcvItemList(altemList, nCount, aValues)
{
    Main.MessageList("검색된종목수",nCount);

    //검색된 종목이 한종목 이상
    if (nCount >= 1)
    {
        //종목객체 요청리스트를 초기화
        MKList = [];

        //처음 종목이 검색이 될 때는
        if (MK.length == 0)
        {
            //검색된 종목 중 잔고에 없는 종목으로 최대20개만 MKList에 저장
            for (var a = 0; a < nCount; a++)
            {
                Account1.SetBalance(altemList[a],0);
                if (Account1.Balance.count == 0 && MKList.length < 최대매수종목수)
                {
                    MKList.push(altemList[a]);
                }
            }
        }
        else
        {
            //1종목이상 검색이 된 이후에는
            if (MK.length < 최대매수종목수)
            {
                for (var a = 0; a < nCount; a++)
                {
                    //종목객체가 만들어진 종목은 제외
                    var Add = true;
                    for (var b = 0; b < MK.length; b++)
                    {
                        if (altemList[a].종목코드 == MK[b].종목코드)
                        {
                            Add = false;
                            break;
                        }
                    }
                    if (Add)
                    {
                        MKList.push(altemList[a]);
                    }
                }
            }
        }
    }
}

```

```

    {
        if (altemList[a] == MK[b].code)
        {
            Add = false;
        }
    }

    //계좌 잔고에 보유중인 종목 제외
    Account1.SetBalance(altemList[a],0);
    if (Account1.Balance.count > 0)
        Add = false;

    //종목객체로 만들어 지지 않았고 잔고에 없는 종목으로 선정하여
    //요청리스트에 저장
    if (Add == true && MKList.length < 최대매수종목수-MK.length)
    {
        MKList.push(altemList[a]);
    }
}

Main.MessageList("주문할 종목수:",MKList.length);

//요청할 종목이 없고
if (MKList.length == 0)
{
    //종목객체가 최대매수종목수 미만이면 1번 타이머 재셋팅
    if (MK.length < 최대매수종목수)
    {
        //1번 타이머, 60초
        Main.SetTimer(1, 6000);
    }
    else
    {
        //종목객체가 최대매수종목수와 같으면 메세지 출력하고 종료
        Main.MessageList("최대종목수 도달 검색종료");
    }
}

```

```

        }

    }

    else //요청할 종목이 있으면
    {

        //요청리스트의 첫종목 부터 종목객체 요청(배열이므로 종목코드가 0번방 부터 존재)
        req = 0;

        Main.MessageList(req, "종목객체요청", MKList[req]);
        Main.ReqMarketData(MKList[req]);

    }

}

//요청한 종목객체가 만들어 지면
function Main_OnRcvMarketData(MarketData)
{
    //직전 요청한 종목이 맞는지 확인후에
    if (MarketData.code == MKList[req])
    {

        Main.MessageList(req, "종목객체생성", MarketData.code);
        //종목객체는 MK에 추가하고
        MK.push(MarketData);

        //매수주문가격
        var OP = MarketData.Ask(1);
        //수량
        var OV = Math.floor(매수금/OP);

        //지정한 가격, 지정한 수량으로 매수주문
        Account1.OrderBuy(MarketData.code, OV, OP, 0);
        Main.MessageList(req, "매수주문", MarketData.code, OV, OP);

        //요청횟수 1증가
        req = req+1;

        //요청횟수가 요청할 종목갯수보다 작으면 다음 종목객체 요청
        if (req < MKList.length)
    }
}

```

```

    }

    Main.MessageList(req,"종목객체요청",MKList[req]);
    var XX = Main.ReqMarketData(MKList[req]);

    //종목객체 생성제한
    if (XX == -1)
    {
        //2번 타이머 15초
        Main.SetTimer(2,15000);
    }
}

else //요청횟수가 요청할 종목갯수에 도달했으면 종목객체 생성 종료
{
    Main.MessageList(req,"종목객체생성끝");
    //최대매수종목수에 도달하지 않았으면 종목검색을 위해 타이머 재셋팅
    if (MK.length < 최대매수종목수)
    {
        Main.SetTimer(1,6000);
    }
    else
    {
        //최대매수종목수를 채웠으면 검색 종료
        Main.MessageList("최대종목수 도달 검색종료");
    }
}
}
}

```

#### 예제14. 종목검색에서 결과값 이용

종목검색화면에서 결과값은 사용자검색조건에 조건식이 1개이면 수식에서 결과값으로 지정한 값이 출력되지만 조건식이 2개 이상이면 조건만족여부만 O,X로 알려주게 됩니다.

[3202] 파워종목검색

변수이름	변수값
Period	12
Period1	5
Period2	5

매수종목검색  검색에 필요한 최소 기간 : 0 봄  
기준봉 0 일 ● 주 ● 월 ● 분 1 설정 초기화 편집

조건 A [매수종목검색(12,5,5)] 사용자 설정 검색기간 사용하지 않음 기준봉:[0] 사용데이터:일봉  
내용 (검색식1) 목록에 조건 추가 결과 삭제  
조건식이 1개인 경우

조건식 A

검색식1		검색	종목 불러오기	종목 내보내기	설정	<input checked="" type="checkbox"/>	상
<input checked="" type="checkbox"/>	종목명(234)	검색결과값	종목코드	현재가	전일대비(%)	거래량	
<input checked="" type="checkbox"/>	(K) BYC우	54.68	001465	143,700	0 ( 0.00 )	0	
<input checked="" type="checkbox"/>	(K) DL건설	53.25	001880	14,350	0 ( 0.00 )	0	
<input checked="" type="checkbox"/>	(Q) GH신소재	59.97	130500	2,800 ▲	15 ( 0.54 )	6,671	
<input checked="" type="checkbox"/>	(Q) IBKS제22호스팩	94.30	448760	2,455 ▲	5 ( 0.20 )	1	
<input checked="" type="checkbox"/>	(K) IHQ	33.33	003560	10,760	0 ( 0.00 )	0	
<input checked="" type="checkbox"/>	(Q) ITX-AI	33.33	099520	1,284	0 ( 0.00 )	0	
<input checked="" type="checkbox"/>	(K) KCTC	24.25	009070	3,985 ▲	5 ( 0.13 )	12,441	
<input checked="" type="checkbox"/>	(Q) KG ETS	21.82	151860	10,790 ▲	310 ( 2.96 )	159,274	
<input checked="" type="checkbox"/>	... 117 건	22.22	226260	210	0 ( 0.00 )	0	

기능순 가나다순 정상 처리 되었습니다.

[3202] 파워종목검색

변수이름	변수값
Period	12
Period1	5
Period2	5

매수종목검색  검색에 필요한 최소 기간 : 0 봄  
기준봉 0 일 ● 주 ● 월 ● 분 1 설정 초기화 편집

조건 A [매수종목검색(12,5,5)] 사용자 설정 검색기간 사용하지 않음 기준봉:[0] 사용데이터:일봉  
B [매도종목검색(12,5,5)] 사용자 설정 검색기간 사용하지 않음 기준봉:[0] 사용데이터:일봉  
내용 (검색식2) 목록에 조건 추가 결과 삭제  
조건식이 2개 이상인 경우

조건식 A or B

검색식2		검색	종목 불러오기	종목 내보내기	설정	<input checked="" type="checkbox"/>	상
<input checked="" type="checkbox"/>	종목명(340)	미결제약정	예상체결가	예상체결량	제가전일종가대비( A B )		
<input checked="" type="checkbox"/>	(K) BYC우	0	0	0	0 ( 0.00 ) ○ × X		
<input checked="" type="checkbox"/>	(K) DL건설	0	0	0	0 ( 0.00 ) ○ × X		
<input checked="" type="checkbox"/>	(Q) DSC인베스트먼트	0	0	0	0 ( 0.00 ) × ○ X		
<input checked="" type="checkbox"/>	(Q) GH신소재	0	0	0	0 ( 0.00 ) ○ × X		
<input checked="" type="checkbox"/>	(K) HDC현대산업개발	0	0	0	0 ( 0.00 ) × ○ X		
<input checked="" type="checkbox"/>	(K) HMM	0	0	0	0 ( 0.00 ) × ○ X		
<input checked="" type="checkbox"/>	(Q) IBKS제20호스팩	0	0	0	0 ( 0.00 ) × ○ X		
<input checked="" type="checkbox"/>	(Q) IBKS제22호스팩	0	0	0	0 ( 0.00 ) ○ × X		
<input checked="" type="checkbox"/>	... 117 건	0	0	0	0 ( 0.00 ) ○ × X		

기능순 가나다순 정상 처리 되었습니다.

스팟에서 종목검색에 리턴되는 결과값도 이와 같습니다.

1개의 조건식으로 구성된 사용자검색조건은 aValues에 종목검색식에서 지정한 값이 리턴되고 결과값은 내림차순으로 정렬되어 제공됩니다.

The screenshot shows a debugger interface with a code editor on the left and a data viewer on the right. The code editor contains the following JavaScript-like pseudocode:

```

1 function Main_OnStart()
2 {
3     Main.MessageList("Start");
4     Main.ReqPowerSearch("검색식1");
5 }
6
7 function Main_OnRcvItemList(aItemList, nCount,aValues)
8 {
9     if (nCount > 0)
10    {
11        for (var i = 0; i < nCount; i++)
12        {
13            Main.MessageList(i,aItemList[i],aValues[i]);
14        }
15    }
16 }
17
18
19
20
21
22

```

The data viewer on the right displays a list of items with numerical values, each preceded by a number from 0 to 21. A yellow box highlights the first few items:

- 0 019175 29.76
- 1 307930 18.46
- 2 089230 17.14
- 3 258830 16.84
- 4 256840 15.04
- 5 041190 14.67
- 6 003535 13.48
- 7 123840 12.82
- 8 294090 12.21
- 9 019170 12.17
- 10 039980 11.96
- 11 131030 10.82
- 12 003530 10.47
- 13 330730 8.82
- 14 036090 8.73
- 15 317330 8.01
- 16 099410 7.94
- 17 352770 7.69
- 18 145720 7.22
- 19 123410 7.06
- 20 095500 6.79
- 21 418550 5.73

2개 이상의 조건식으로 구성된 사용자검색조건은 aValues에 만족여부만 O,X대신에 1,0으로 제공됩니다.

This screenshot shows the same debugger interface with the same code editor content. The data viewer on the right displays a list of items with binary search results, each preceded by a number from 0 to 21. A yellow box highlights the first few items:

- 0 900340 1,1
- 1 000040 1,0
- 2 000087 1,0
- 3 000100 1,0
- 4 000105 1,0
- 5 000250 1,1
- 6 000270 1,0
- 7 000300 1,0
- 8 000520 1,0
- 9 000640 0,1
- 10 000660 1,0
- 11 000760 1,0
- 12 000860 1,0
- 13 000970 1,0
- 14 001065 1,0
- 15 001067 1,0
- 16 001130 1,0
- 17 001200 1,0
- 18 001275 1,0
- 19 001360 1,0
- 20 001430 1,0
- 21 001460 1,0

검색 결과값은 스트링으로 2개이상의 조건으로 이루어진 경우 결과값이 콤마로 구분되어 있는 스트링입니다.

그러므로 조건을 지정할 경우 “1,0”와 같이 지정해 조건을 지정해야 합니다.

```

1  function Main_OnStart()
2  {
3      Main.MessageList("Start");
4      Main.ReqPowerSearch("검색식2");
5  }
6
7
8  function Main_OnRcvItemList(aItemList, nCount,aValues)
9  {
10     Main.MessageList(nCount);
11     if (nCount > 0)
12     {
13         for (var i = 0; i < nCount; i++)
14         {
15             Main.MessageList(i,aItemList[i],aValues[i]);
16
17             //사용자검색조건 중 A가 만족한 경우 매수
18             if (aValues[i] == "1,0")
19                 Account1.OrderBuy(aItemList[i], 1, 0, 1);
20
21             //사용자검색조건 중 B가 만족한 경우 매도
22             if (aValues[i] == "0,1")
23                 Account1.OrderSell(aItemList[i], 1, 0, 1);
24
25         }
26     }
27 }

```

만약 각 조건의 만족여부값을 각각 지정하고자 하면

JSON.parse 함수로 값을 변환해서 이용하시면 됩니다.

```

1  function Main_OnStart()
2  {
3      Main.MessageList("Start");
4      Main.ReqPowerSearch("검색식2");
5  }
6
7
8  function Main_OnRcvItemList(aItemList, nCount,aValues)
9  {
10     Main.MessageList(nCount);
11     if (nCount > 0)
12     {
13         for (var i = 0; i < nCount; i++)
14         {
15             Main.MessageList(i,aItemList[i],aValues[i]);
16
17             //사용자검색조건 중 A가 만족한 경우 매수
18
19             var AB = JSON.parse("[ "+aValues[i]+"]");
20
21             if (AB[0] == 1)
22                 Account1.OrderBuy(aItemList[i], 1, 0, 1);
23
24             //사용자검색조건 중 B가 만족한 경우 매도
25             if (AB[1] == 1)
26                 Account1.OrderSell(aItemList[i], 1, 0, 1);
27
28         }
29     }
30 }
31

```

### ※ 예스매매신호의 결과값

예스랭귀지로 작성한 종목검색식은 find(출력결과값); 으로 작성되어 1개의 결과값만 출력이 가능합니다.

하지만 종목검색창에서 제공되는 예스매매신호들 중 일부는 여러 개의 항목을 출력하게 되어 있습니다.

종목명(51)	연평균손익	승률	거래횟수(수익/손실)	평균수익/평균손실	현상태	종목코드
(K) GS건설	2.73%	30.43%	23(7/16)	2.64	매수신호	006360
(K) HDC현대산업개발	-1.81%	45.00%	20(9/11)	1.08	매수신호	294870
(K) HD현대인프라코어	11.14%	32.79%	61(20/39)	2.38	매수신호	042670
(K) KCTC	15.72%	45.71%	35(16/19)	1.93	매수신호	009070
(Q) NEW	40.37%	38.64%	44(17/27)	2.40	매수신호	160550
(K) S-Oil	8.42%	42.03%	69(29/40)	1.80	매수신호	010950
(K) SIMPAC	43.32%	41.03%	78(32/46)	2.56	매수신호	009160
(Q) THE E&M	24.55%	32.43%	37(12/24)	2.80	매수신호	089230
(Q) TJ미디어	8.05%	32.18%	87(28/59)	2.52	매수신호	032540
(O) 골드에스	20.63%	33.33%	24(8/16)	3.78	매수신호	035290

이에 따라 사용자검색조건에 예스매매신호를 사용하면 리턴되는 결과값은

"현재상태,총거래횟수,수익거래횟수,손실거래횟수,연평균손익,승률,평균수익/평균손실"

와 같이 각 종목별로 여러 개의 값이 콤마로 구분되어 스트링으로 제공되고

연평균손익기준으로 내림차순 정렬되어 제공됩니다.

그러므로 예스매매신호의 각 항목의 값을 이용하고자 하면

JSON.parse 함수로 값을 변환해서 이용하시면 됩니다.

```

1 function Main_OnStart()
2 {
3     Main.MessageList("Start");
4     Main.ReqPowerSearch("검색식1");
5 }
6
7
8 function Main_OnRcvItemList(aItemList, nCount,aValues)
9 {
10     Main.MessageList(nCount);
11     if (nCount > 0)
12     {
13         for (var i = 0; i < nCount; i++)
14         {
15             var AB = JSON.parse("[ "+aValues[i]+"]");
16
17             Main.MessageList(aItemList[i],"현재상태:",AB[0],"총거래횟수:",AB[1],"수익거래횟수:",AB[2],
18                             "손실거래횟수:",AB[3],"연평균손익:",AB[4],"승률:",AB[5],"평균수익/평균손실:",AB[6]);
19             //현재상태는 '0':매매신호없음 '1':매수신호 '2':매수유지 '3':매도신호 '4':매도유지 '5':포지션미보유 '6':손절매신호
20
21         }
22     }
23 }
24
25 디버깅
26 출력보기선택: 모두
27 Start
28 51
29 222810 현재상태: 1 총거래횟수: 79 수익거래횟수: 30 손실거래횟수: 48 연평균손익: 666.546258 승률: 37.974684 평균수익/평균손실: 3.924109
30 247540 현재상태: 1 총거래횟수: 54 수익거래횟수: 28 손실거래횟수: 23 연평균손익: 390.53619 승률: 51.851852 평균수익/평균손실: 2.303678
31 063080 현재상태: 1 총거래횟수: 56 수익거래횟수: 22 손실거래횟수: 33 연평균손익: 142.450974 승률: 39.285714 평균수익/평균손실: 3.769999
32 194480 현재상태: 1 총거래횟수: 83 수익거래횟수: 34 손실거래횟수: 48 연평균손익: 94.056035 승률: 40.963855 평균수익/평균손실: 1.983656
33 261200 현재상태: 1 총거래횟수: 29 수익거래횟수: 10 손실거래횟수: 19 연평균손익: 82.141308 승률: 34.482759 평균수익/평균손실: 4.450084
34 091440 현재상태: 1 총거래횟수: 45 수익거래횟수: 20 손실거래횟수: 23 연평균손익: 60.848686 승률: 44.444444 평균수익/평균손실: 2.534465
35 086900 현재상태: 1 총거래횟수: 36 수익거래횟수: 16 손실거래횟수: 20 연평균손익: 58.969291 승률: 44.444444 평균수익/평균손실: 4.49041
36 391710 현재상태: 1 총거래횟수: 14 수익거래횟수: 6 손실거래횟수: 7 연평균손익: 48.801367 승률: 42.857143 평균수익/평균손실: 2.576078
37 348210 현재상태: 1 총거래횟수: 22 수익거래횟수: 11 손실거래횟수: 11 연평균손익: 45.348055 승률: 50 평균수익/평균손실: 2.035667
38 002700 현재상태: 1 총거래횟수: 49 수익거래횟수: 24 손실거래횟수: 25 연평균손익: 44.843018 승률: 48.979592 평균수익/평균손실: 2.313945
39 019490 현재상태: 1 총거래횟수: 35 수익거래횟수: 14 손실거래횟수: 21 연평균손익: 44.582827 승률: 40 평균수익/평균손실: 2.931396
40 095500 현재상태: 1 총거래횟수: 60 수익거래횟수: 26 손실거래횟수: 34 연평균손익: 44.026787 승률: 43.333333 평균수익/평균손실: 2.267863
41 009160 현재상태: 1 총거래횟수: 78 수익거래횟수: 32 손실거래횟수: 46 연평균손익: 43.320207 승률: 41.025641 평균수익/평균손실: 2.561288
42 036930 현재상태: 1 총거래횟수: 53 수익거래횟수: 23 손실거래횟수: 30 연평균손익: 40.45688 승률: 43.396226 평균수익/평균손실: 2.144075
43 160550 현재상태: 1 총거래횟수: 44 수익거래횟수: 17 손실거래횟수: 27 연평균손익: 40.36848 승률: 38.636364 평균수익/평균손실: 2.39929
44 104700 현재상태: 1 총거래횟수: 29 수익거래횟수: 19 손실거래횟수: 10 연평균손익: 40.062028 승률: 65.517241 평균수익/평균손실: 2.317463

```

디버깅 | 찾기결과1 | 찾기결과2