

# Example ACSIL Trading Systems

---

- [Introduction](#)
  - [Introduction](#)
- 

## Introduction

---

This page lists example ACSIL (Advanced Custom Study Interface and Language) trading systems. For additional information, refer to [Automated Trading From an Advanced Custom Study](#).

There are also many example trading systems in the `/ACS_Source/TradingSystem.cpp` file in the folder where Sierra Chart is installed to on your system.

## Trading Example 1

---

### Example

```
#include "sierrachart.h"

SCDLLName("SCTradingExample1")

/*
Overview
-----
An example of a trading system that enters a new position or
reverses an existing one on the crossover of two study Subgraphs.

When line1 crosses line2 from below the system will go long.

When line1 crosses line2 from above, the system will go short.

Comments
-----
* Let the user of this trading system study select the two study
Subgraph lines to monitor for a crossover. This is accomplished using
sc.Input[].SetStudySubgraphValues. In the Study Settings the user is
provided with list boxes to select the study and subgraph.

* The example uses the Auto Trade Management reversal functionality by
setting sc.SupportReversals to 1. This means that we simply call
sc.BuyEntry for a long and sc.SellEntry for a short with the number of
contracts we want to long/short. In the example the number of contracts
is 1.

So:
** If we are flat, we will enter 1 contract long/short.
** If we are currently short, Sierra Chart will reverse the position
for us and we will be 1 long.
** If we are currently long, Sierra Chart will reverse the Position
for us and we will be 1 short.
```

\* For the simplicity of the example, the study process events on the close of the bar.

\* To keep the example simple, the study uses market order types to enter the Position.

\* Note that if the system enters a Position, and the user manually closes the Position, the system will remain flat until the next crossover at which point a new Position will be established.

\*/

```
SCSFExport scsf_SC_TradingCrossOverExample(SCStudyInterfaceRef sc)
{

    SCInputRef Input_Subgraph1Reference = sc.Input[0];
    SCInputRef Input_Subgraph2Reference = sc.Input[1];

    SCSubgraphRef Subgraph_BuyEntry = sc.Subgraph[0];
    SCSubgraphRef Subgraph_SellEntry = sc.Subgraph[1];

    if (sc.SetDefaults)
    {

        // Set the configuration and defaults

        sc.GraphName = "Trading CrossOver Example";

        sc.StudyDescription = "An example of a trading system that enters a new position or \
                                reverses an existing one on the crossover of two study Subgraphs. \
                                When Subgraph1Reference crosses Subgraph2Reference from below, the system will l
                                When Subgraph1Reference crosses Subgraph2Reference from above, the system will l

        sc.AutoLoop = 1; // true
        sc.GraphRegion = 0;
        sc.CalculationPrecedence = LOW_PREC_LEVEL;

        Input_Subgraph1Reference.Name = "Line1";
        Input_Subgraph1Reference.SetStudySubgraphValues(1, 0);

        Input_Subgraph2Reference.Name = "Line2";
        Input_Subgraph2Reference.SetStudySubgraphValues(1, 0);

        Subgraph_BuyEntry.Name = "Bullish";
        Subgraph_BuyEntry.DrawStyle = DRAWSTYLE_POINT_ON_HIGH;
        Subgraph_BuyEntry.LineWidth = 3;

        Subgraph_SellEntry.Name = "Bearish";
        Subgraph_SellEntry.DrawStyle = DRAWSTYLE_POINT_ON_LOW;
        Subgraph_SellEntry.LineWidth = 3;

        sc.AllowMultipleEntriesInSameDirection = false;
        sc.MaximumPositionAllowed = 5;
        sc.SupportReversals = true;

        // This is false by default. Orders will go to the simulation system always.
        sc.SendOrdersToTradeService = false;

        sc.AllowOppositeEntryWithOpposingPositionOrOrders = false;
        sc.SupportAttachedOrdersForTrading = false;

        sc.CancelAllOrdersOnEntriesAndReversals = true;
        sc.AllowEntryWithWorkingOrders = false;
        sc.CancelAllWorkingOrdersOnExit = true;
        sc.AllowOnlyOneTradePerBar = true;

        sc.MaintainTradeStatisticsAndTradesData = true;
```

```

return;
}

// only process at the close of the bar; if it has not closed don't do anything
if (sc.GetBarHasClosedStatus() == BHCS_BAR_HAS_NOT_CLOSED)
{
return;
}

// using the Input_Subgraph1Reference and Input_Subgraph2Reference input variables,
// retrieve the subgraph arrays into Subgraph1Reference, Subgraph2Reference arrays respectively
SCFloatArray Subgraph1Reference;
sc.GetStudyArrayUsingID(Input_Subgraph1Reference.GetStudyID(), Input_Subgraph1Reference.GetSubg

SCFloatArray Subgraph2Reference;
sc.GetStudyArrayUsingID(Input_Subgraph2Reference.GetStudyID(), Input_Subgraph2Reference.GetSubg

// code below is where we check for crossovers and take action accordingly

if (sc.CrossOver(Subgraph1Reference, Subgraph2Reference) == CROSS_FROM_BOTTOM)
{
// mark the crossover on the chart
Subgraph_BuyEntry[sc.Index] = sc.Low[sc.Index];

// Create a market order and enter long.
s_SCNewOrder NewOrder;
NewOrder.OrderQuantity = 1;
NewOrder.OrderType = SCT_ORDERTYPE_MARKET;
NewOrder.TimeInForce = SCT_TIF_GOOD_TILL_CANCELED;

sc.BuyEntry(NewOrder);
}

if (sc.CrossOver(Subgraph1Reference, Subgraph2Reference) == CROSS_FROM_TOP)
{
// mark the crossover on the chart
Subgraph_SellEntry[sc.Index] = sc.High[sc.Index];

// create a market order and enter short
s_SCNewOrder NewOrder;
NewOrder.OrderQuantity = 1;
NewOrder.OrderType = SCT_ORDERTYPE_MARKET;
NewOrder.TimeInForce = SCT_TIF_GOOD_TILL_CANCELED;

sc.SellEntry(NewOrder);
}
}

```

\*Last modified Wednesday, 22nd February, 2023.