Jurik Research Limited Use Software License Agreement

CONCERNING THE SOFTWARE AND DOCUMENTATION, THIS LICENSE AGREEMENT IS THE ENTIRE AGREEMENT BETWEEN JURIK RESEARCH AND YOU. IN ORDER TO HAVE COMPLETED INSTALLATION OF THIS SOFTWARE, YOU GAVE CONSENT TO BE BOUND BY THE FOLLOWING TERMS AND CONDITIONS. KEEP THIS AGREEMENT WITH YOUR PERMANENT RECORDS.

PREFACE -- This manual (the "Documentation") refers to commercial software products (the "Software") provided by Jurik Research ("JR"). This Software will not operate unless you purchase or already own a fully paid license from JR. JR licenses its use under the terms set forth herein.

LICENSE GRANT -- If you are a fully paid license holder, Jurik Research, as licensor, grants to you, and you accept, a non-exclusive license to use the enclosed program Software and accompanying Documentation, only as authorized in this agreement. You acquire no right, title or interest in or to the Documentation or Software, whose copyrights are owned by Jurik Research (JR). All rights are reserved. You agree to respect and to not remove or conceal from view any copyright notices appearing on the Software or Documentation. You may not sublicense this license. You may not rent, lease, modify, translate, disassemble, decompile, or reverse engineer the software, nor create derivative works based on the software without written permission from JR. The software may be used only on a single computer at a single location at any one time. JR permits you to make one archival copy of the software. No other copies or any portions thereof may be made by you or by any persons under your control. No part of this manual may be transmitted or reproduced in any form, by any means, for any purpose other than the purchaser's personal use without written permission of JR.

TERM -- This license is effective until terminated. You may terminate it at any time. It will also terminate upon conditions set forth elsewhere in this Agreement if you fail to comply with any term or condition of this Agreement. You agree upon such termination to destroy the Software and Documentation together with all copies, modifications and merged portions of the software expressed in any media form, including archival copies.

LIMITED WARRANTY -- The information in the user guide and on the diskette is subject to change without notice and does not represent a commitment on the part of JR. JR warrants the diskette and physical document to be free of defects in materials and workmanship. The user's sole remedy, in the event a defect is found, is that JR will replace the defective diskette or documentation. JR warrants that the Software, if properly installed and operated on software for which it is designed, will perform substantially as described in the Documentation. JR also warrants the software to be operationally compatible with the software platforms and operating systems as specified on the JR website, whose software version numbers for each relevant software product are also specified at said website. You recognize and accept that there is the possibility that a software platform developer or operating system developer may significantly change their product so as be incompatible with Jurik tools. Although JR may create a revised version of its tools to re-establish compatibility, no warranty is expressed or implied to that effect. In the case said incompatibility does occur, JR is under no obligation to provide a refund, product exchange, revision or upgrade. The above express warranty is the only warranty made by JR. It is in lieu of any other warranties, whether expressed or implied, including, but not limited to, any implied warranty of merchantability of fitness for a particular purpose. This warranty commences on the date of delivery to the Licensee and expires sixty (60) days thereafter. Any misuse or unauthorized modification of the Software will void this limited warranty. No JR dealer, distributor, agent or employee is authorized to make any modification or addition to this warranty. This warranty gives you specific legal rights, and you may have other rights that vary from state to state. Product may not be returned for a refund after warranty has expired.

LIMITATION OF LIABILITY -- Because computer software is inherently complex and may not be completely free of errors, you are advised to test the software thoroughly before relying upon it. JR will not be responsible for your failure to do so. You assume full responsibility and risk for Software installation, application and results. Do not use the Software in any case where an error may cause significant damage or injury to persons, property or business. In no event shall JR be liable for any indirect, special, incidental, tort, economic, cover, consequential, exemplary damages or other damages, regardless of type, including, without limitation, damages or costs relating to the loss of profits, business, goodwill, data, or computer programs, arising out of the use of or inability to use JR products or services, even if the company or its agent has been advised of the possibility of such damages or of any claim by any other party. JR's total liability to you or any other party for any loss or damages resulting from any claims, demands or actions arising out of or related to this agreement shall not exceed the license fee paid to JR for use of this software. Some states or provinces do not allow the exclusion or limitation of implied warranties or limitation of liability for incidental or consequential damages, so the above exclusion or limitation may not apply to you.

GOVERNING LAW and COST OF LITIGATION -- The license agreement shall be construed and governed in accordance with the laws of California. In any legal action regarding the subject matter hereof, venue shall be in the state of California, and the prevailing party shall be entitled to recover, in addition to any other relief granted, reasonable attorney fees and expenses of litigation. The export of JR products is governed by the U.S. Department of Commerce under the export administration regulations. It is your responsibility to comply with all such regulations.

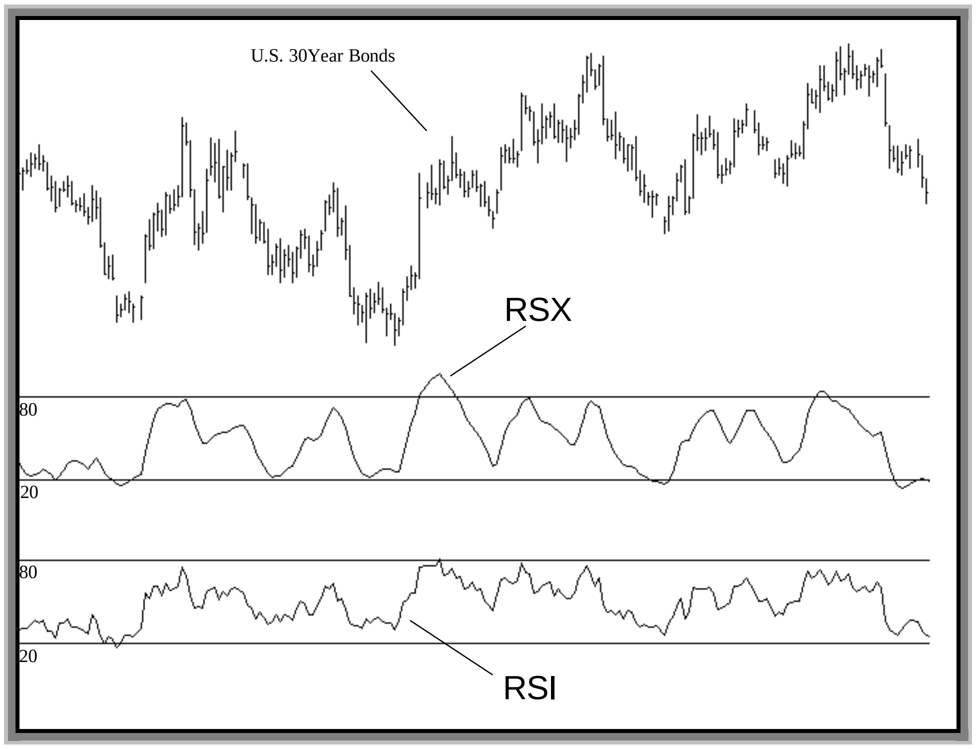
NO WAIVER -- The failure of either party to enforce any rights granted hereunder shall not be deemed a waiver by that party as to subsequent enforcement of rights in the event of future breaches.

FEES -- A new password is required for installing Software into each additional computer. This license entitles you up to two passwords, one for each computer that you own. There is a fee for each additional password beyond the first two. Violation of this restriction is a direct copyright infringement to which Jurik Research shall seek legal remedy. Future upgrades may require a fee. Prices may change without notice.

JURIK RESEARCH

RSX

Relative Trend Strength Index DLL module for Windows® Application Developers



USER’S GUIDE

Requirements

- Windows 98, 2000, NT4 or XP
- Application software that can access DLL functions.

Installing the 32 bit DLL module

1. Execute the Installer, JRS_DLL.EXE. It will analyze your computer and give you a computer identification number. Write it down.
2. Get your access PASSWORD from Jurik Research Software. You can do so by calling 323-258-4860 (USA), faxing 323-258-0598 (USA), e-mailing support@nfsmith.net, or writing Jurik Research Software at 686 South Arroyo Parkway, Suite 237, Pasadena, California 91105. Be sure to give your full name, mailing address and computer identification number. You will then be given a password.
3. Rerun the installer JRS_DLL.EXE, this time entering the password when asked. Also enter **all the Jurik Research modules that you currently are licensed to run**. It will copy the latest version of these modules to any directory you specify.

You may now code your software to access the DLL as described on the following pages. First, read the important notices below.

!! IMPORTANT !!

ABOUT PASSWORDS
And what to do when they become invalid

If you upgrade to a new computer, or significantly upgrade your existing computer (such as flash a new BIOS), you should reinstall RSX and all other Jurik tools that are licensed for your computer. The installer will let you know if your current password is no longer valid. Also, if you want to run RSX on additional computers, you will need additional passwords. For new or replacement passwords, call 323-258-4860

ABOUT DATA VALIDITY
And what to do when RSX encounters an error

When RSX encounters a problem, (e.g. the password used during installation has become invalid), RSX will continue to run but the data produced will not be valid. To let you know this is the case, RSX will return an appropriate error code, bit it will NOT post any warning message on your monitor. Therefore …

Do not assume RSX results are correct. You must validate RSX's output by CHECKING THE RETURN ERROR CODE immediately after each call to RSX.

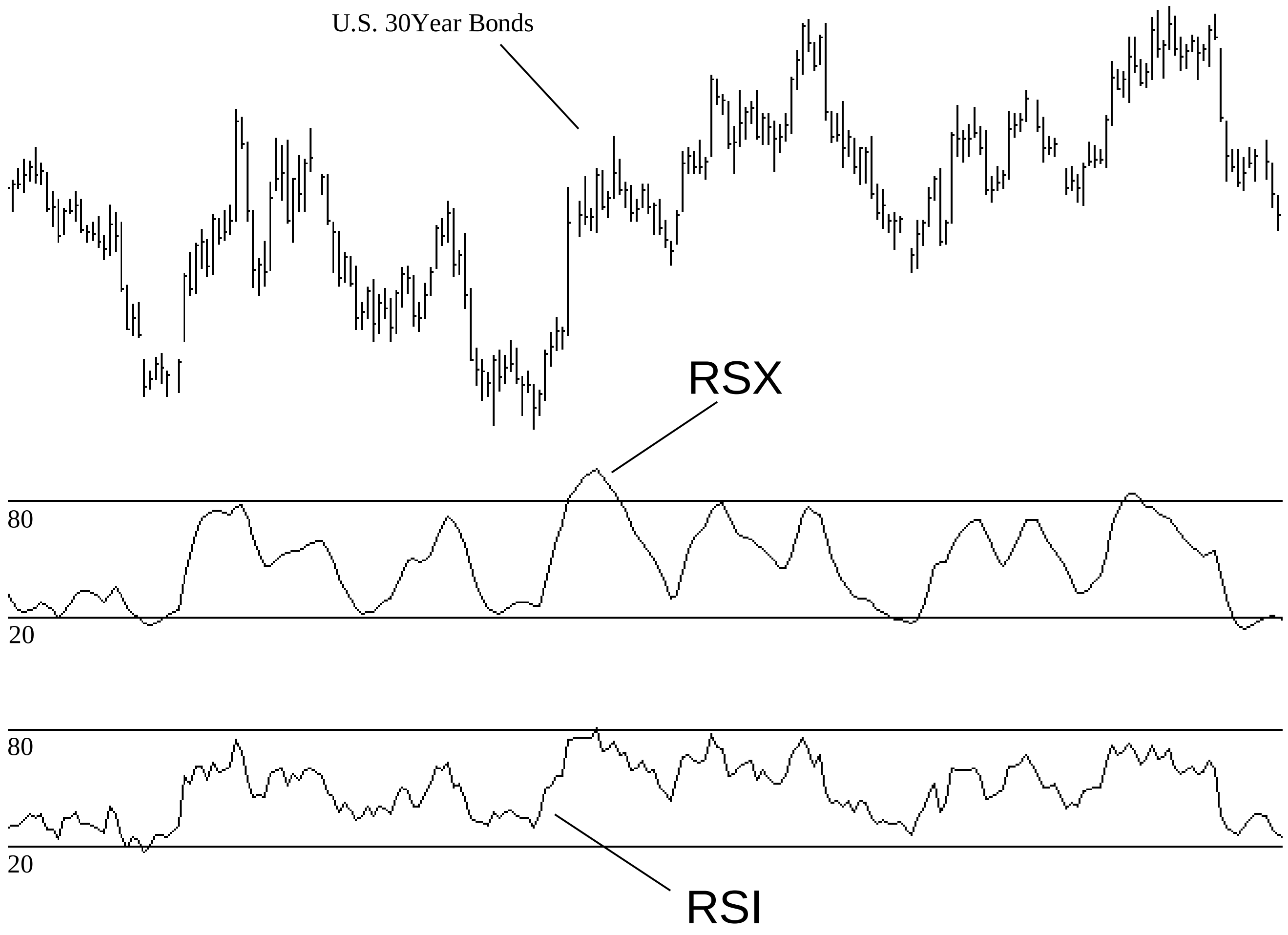
Why use **RSX ?**

The popular RSI indicator is very noisy.   RSX eliminates noise completely !!

There is only one convincing way to illustrate the power of RSX. In the chart below, we see daily bars of U.S. Bonds analyzed by RSX and the classical RSI.

RSX is very smooth. Typically any indicator can be smoothed by a moving average, but the penalty is added lag to the resulting signal. Not only is RSX smoother than RSI, but its smoothness comes ***without added lag***.

RSX permits more accurate analysis, helping you avoid many trades that would have been prematurely trigged by the jagged RSI. Once you begin using RSX, you may never apply the classical RSI again!



Coding Applications

The DLL file contains two versions of **RSX**.

The **BATCH MODE** version accepts an entire array of input data and returns results into another array of equal size. This method requires ther user provide the DLL function with pointers to two arrays. This version is ideal when an entire array is available for processing with only one call to RSX.

The **REAL TIME** version accepts one input value and returns one value as a result. RSX is called for each successive value in some arbitrary time series. This approach is ideal for processing real time data, whereby the user wants an instant RSX update as each new data value arrives.

The following pages cover the following applications of RSX:

- C code example for batch mode
- C code example for real time
- Visual Basic example for batch mode
- Visual Basic example for real time

Dynamic Linking

Load Time Dynamic Linking (Microsoft Compilers)

For load-time dynamic linking, you must use the LIB file JRS_32.LIB, located at C:\JRS_DLL\LIB (or on whichever drive you specified during installation). With load-time dynamic linking, the Jurik DLL is loaded into memory when the user's EXE is loaded.

Load Time Dynamic Linking (non-Microsoft Compilers)

The LIB file we provide will only work with the MS Visual C/C++ compiler. For C/C++ users with non-Microsoft compilers, you will probably not be able to use the LIB file we have provided for Load Time Dynamic Linking with our DLL functions. You have two choices. 1) Consult your compilers' documentation to determine how to construct a LIB file from a DLL. For instance, Borland's compiler includes the IMPLIB.EXE utility to accomplish this. 2) Use run-time dynamic linking (described below). A LIB file is not required for this method.

Run Time Dynamic Linking

You may prefer to use run-time dynamic linking instead of load-time. For example, users of Microsoft Visual C may wish to prevent the Jurik DLL from automatically loading along with the user's EXE. With run-time, the DLL is loaded only when the user's EXE specifically calls for it to be loaded with the LoadLibrary function. Another reason for prefering run-time is that the user has a non-Microsoft compiler, and therefore, cannot use the LIB file provided.

For new C/C++ users, we provide sample C files which demonstrate how to accomplish run-time dynamic linking. The sample files are located in the folder C:\JRS_DLL\RUNTIME (or on whichever drive you specified during installation).

C Programming the 32 bit RSX DLL for batch mode

The file **JRS_32.DLL** contains the function RSX. In your C code, you should declare RSX as externally defined and, if using MS VC++, use the _declspec(dllimport) keywords. The function is exported as a C function, so if you are using C++, you should insert "C" (with the quotes) between the words "extern" and "_declspec". Also, you should link with JRS_32.LIB, which we provide.

```
extern _declspec(dllimport) int WINAPI RSX( DWORD iSize,
                    double *pdSeries, double *pdOutput, double dSmooth );
```

PARAMETERS

iSize: A 32 bit unsigned integer equal to the number of doubles in the input data array.
pdSeries: A pointer to an array of doubles that contain your input time series data for RSX.
pdOutput: A pointer to an array of doubles that RSX will write its results to.
dSmooth: A double precision floating point number that controls the smoothness of RSX's curve.

NOTES

Both input and output arrays must be of the same size, as specified by the calling parameter **iSize**.

dSmooth may be any value between 2 and 500 inclusive. Typical smoothness values range from 5 - 20.

Although RSX reads all the input data, it does not attempt to produce output for the first 29 elements of the input array. This is because RSX needs at least 30 elements to begin its statistical analysis of the data. Consequently, RSX simply outputs the value 50 for this range. True RSX output begins with the 30th element.

RETURN VALUES

The RSX function returns an integer, which will indicate success or an error as below:

0	NO ERROR
-1	PASSWORD / INSTALLATION ERROR
10010	POINTER TO INPUT DATA LOCATION IS NULL
10011	POINTER TO OUTPUT DATA LOCATION IS NULL
10012	NOT ENOUGH DATA ROWS; MUST BE AT LEAST 32
10014	LENGTH PARAMETER BELOW 2
10016	LENGTH PARAMETER ABOVE 500
10017	OUT OF MEMORY

PROGRAMMING EXAMPLE

```
iSize    = 2500;
dSmooth  = 10;
pdSeries = (double *) GlobalAllocPtr( GHND, sizeof(double) * iSize);
pdOutput = (double *) GlobalAllocPtr( GHND, sizeof(double) * iSize);

/* At this location in code, fill up your input array */

error_code = RSX( iSize, pdSeries, pdOutput, dSmooth );
```

C Programming the 32 bit RSX DLL for real time

The file **JRS_32.DLL** contains the function **RSXRT**. In your C code, you should declare RSXRT as externally defined and, if using MS VC++, use the _declspec(dllimport) keywords. The function is exported as a C function, so if you are using C++, you should insert "C" (with the quotes) between the words "extern" and "_declspec". Also, you should link with JRS_32.LIB, which we provide.

```
extern _declspec(dllimport) int WINAPI RSXRT( double dSeries, double
  dSmooth, double *pdOutput, int iDestroy, int *piSeriesID );
```

PARAMETERS

dSeries:	a double precision floating point number equal to the input data value.
dSmooth:	A double precision floating point number that controls the smoothness of RSX's curve.
pdOutput:	a pointer to the memory location of a double which contains the result from RSX
iDestroy:	a 32 bit signed integer, with a value = 0 or 1. When value = 1, the RAM in the DLL used for a particular RSX time series is released. The desired series is designated by piSeriesID. (see next parameter) This event does not release the memory containing the output of RSX. Control of that memory is the user's responsibility.
piSeriesID:	a pointer to the memory location of a 32 bit signed integer (iSeriesID). When processing the first element of any new time series, set iSeriesID = 0. RSX will store a unique identification number of the series into that integer (i.e. iSeriesID) pointed to by pointer piSeriesID.

NOTES

dSmooth may be any value between 2 and 500 inclusive. Typical smoothness values range from 5 - 20.

Although RSX reads all the input data, it does not attempt to produce output for the first 29 times it is called. This is because RSX needs at least 30 input elements to begin its statistical analysis of the data. Consequently, RSX simply outputs the value 50 for the first 29 calls. True RSX output begins with the 30th call.

RETURN VALUES

The RSX function returns an integer, which will indicate success or an error:

0	NO ERROR
-1	PASSWORD / INSTALLATION ERROR
10011	POINTER TO OUTPUT DATA LOCATION IS NULL
10014	LENGTH PARAMETER BELOW 2
10015	POINTER TO SERIES IDENTIFICATION VARIABLE WAS NULL
10016	LENGTH PARAMETER ABOVE 500
10018	CANNOT DEALLOCATE DLL RAM WHEN SERIESID = 0

C PROGRAMMING EXAMPLE for real-time mode

```
// declare variables
double *pdData, *pdOutput, dSmooth ;
int    iDestroy, iSeriesID, *piSeriesID, iErr, i ;

// get address of variable iSeriesID
piSeriesID = &iSeriesID ;

// assume you want this RSX parameter value
dSmooth = 10 ;

// allocate RAM for input and output. Assume array size is 100
pdData   = (double *) GlobalAllocPtr(GHND, sizeof(double) * 100) ;
pdOutput = (double *) GlobalAllocPtr(GHND, sizeof(double) * 100) ;

// fill pdData array with double precision numbers from disk
// file or other source. (code not shown)

// clear deallocation flag and initialize series identification to 0.
iDestroy = iSeriesID = 0 ;

// loop through data, calling RSX on each element, and store results
for(i=0;i<100;i++)
    {
   iErr = RSXRT( *(pdData+i), dSmooth, (pdOutput+i), iDestroy, piSeriesID) ;
   if(iErr != 0)
        YourErrHandlerFunc() ;
    }

// done processing. Deallocate DLL RAM, and check for any errors
// When deallocating, it is OK to replace the output pointer with 0.

iDestroy = 1 ;
iErr = RSXRT( 0,0,0, iDestroy, piSeriesID) ;
if(iErr != 0)
     YourErrHandlerFunc() ;


// do something with data and deallocate RAM at pdData and pdOutput
```

Visual Basic example of RSX in batch mode

INTRODUCTION

In your Jurik Research DLL installation directory (eg., C:\JRS_DLL) the workbook RSX_DLL.XLS contains a programming example using Excel's VBA to call function **RSX**. The workbook includes a worksheet where you can run the macro **RSX_Test** to run **RSX** in batch mode.

In this example, run the VBA macro called “**RSX_Test**”. The macro gets the data in column 1 and sends it to the RSX batch mode function in the DLL. The output array produced by RSX is then written back onto column 3 of the worksheet.

VBA MACRO DESCRIPTION

The macro RSX_TEST calls the function RSX, which is declared as shown below. Note that the input and output arrays (dInData and dOutData) are called by reference using “ByRef”. This enables the calling statement to send to RSX a pointer to the first element of each data array.

```
    Declare Function RSX Lib "JRS_32.dll" ( _
                            ByVal iSize As Long, _
                            ByRef dInData As Double, _
                            ByRef dOutData As Double, _
                            ByVal dLength As Double) As Long
```

The VBA subroutine **RSX_Test** is shown on the next page. This code will read data from column 1 of the active worksheet, call the DLL function RSX, and output its results back to the worksheet.

Note that the code calls a local subroutine “Error_handler”. If an error condition exists, the subroutine posts a message on the screen (because RSX itself does not) and then halts the program.

```
Sub RSX_test()

    Dim k As Long                       'iteration variable
    Dim iSize As Long                   'size of data array
    Dim iResult As Long                 'returned error code
    Dim dInputData() As Double          'input array
    Dim dOutputData() As Double         'output array
    Dim dLength As Double               'RSX speed (smoothness)
    Dim calctype As Long                'for preserving current Excel calc mode

    'disable automatic calculation
    calctype = Application.Calculation
    Application.Calculation = xlManual

    iSize = 100         ' length of input array
    dLength = 10        ' RSX smoothness factor

    ReDim dInputData(1 To iSize)
    ReDim dOutputData(1 To iSize)

    ' Read Data from spreadsheet into array
    ' Input data is in column 1

    For k = 1 To iSize
        dInputData(k) = Cells(k + 1, 1)
    Next k

    '--- RSX return error codes ---
    '    0       SUCCESS -- no error conditions found
    '   -1       password/installation error. RSX output not valid.
    '10012       not enough data rows, must be at least 32
    '10014       length parameter below 2
    '10016       length parameter above 500
    '10017       out of memory

    'Call RSX.  Note that only the first elements of both data arrays are referenced.

    iResult = RSX(iSize, dInputData(1), dOutputData(1), dLength)
    If (iResult <> 0) Then
        Call Error_handler(iResult, calctype)
    Else
        ' Show results in column 3 on spreadsheet
        For k = 1 To iSize
            Cells(1 + k, 3).FormulaR1C1 = dOutputData(k)
        Next k
    End If

    ' Enable automatic calculation
    Application.Calculation = calctype
End Sub
```

```
' The following subroutine is a simple way to handle run-time errors that may occur
' It is good practice to handle each error type mentioned in the user manual.

Private Sub Error_handler(ByVal error_code As Long, ByVal calctype As Long)
    Dim result As Long
    result = MsgBox("Error number " & Str(error_code) & " was returned by RSX.", ,
"RSX Error")
    Application.Calculation = calctype
    End   ' this END command will halt execution of the VBA code.
End Sub
```

Visual Basic example of RSX in real time

INTRODUCTION

In your Jurik Research DLL installation directory (eg., C:\JRS_DLL) the workbook RSX_DLL.XLS contains a programming example using Excel's VBA to call function **RSXRT**. The workbook includes a worksheet where you can run the macro **RSXRT_Test** to run **RSXRT** in real-time mode.

In this example, run the VBA macro called "**RSXRT_Test**". The macro reads one element at a time from column 1, sequentially feeding each one through the real time version of RSX and places the results sequentially into column 4.

VBA MACRO DESCRIPTION

The function RSXRT is declared as shown below. Note that the output and series identification variables (dOutput and iSeriesID) are called by reference using "ByRef". The user intializes the series identification variable to zero and during the first call to RSXRT, the function will replace zero with an integer that uniquely identifies the time series. This way, when you have multiple time series running in parallel, the series identification numbers will tell RSXRT to which time series the new data point is to be assigned.

```
Declare Function RSXRT Lib "JRS_32.dll" ( _
                        ByVal dSeries As Double, _
                        ByVal dLength As Double, _
                        ByRef dRSXout As Double, _
                        ByVal iDestroy As Long, _
                        ByRef iSeriesID As Long) As Long
```

The VBA subroutine **RSXRT_Test** is shown on the next page. This code reads data from column 1 of the active worksheet, one element at a time, each time calling the DLL function RSXRT, and outputing the result back to the worksheet.

Note that the code calls a local subroutine "`Error_handler`". If an error condition exists, the subroutine posts a message on the screen (because RSX itself does not) and then halts the program.

Also note that if you have several separate data time series that you want RSX to process simultaneously in real time, **each time series must be given its own series identification variable**. In this example, only one time series will be filtered, therefore only one series identification variable needs to be declared.

```
Sub RSXRT_test()

    Dim k As Long               'iteration variable
    Dim dLength As Double       'RSX speed (smoothness)
    Dim dRSXout As Double       'RSX output
    Dim iResult As Long         'returned error code
    Dim iDestroy As Long        'deallocate DLL RAM switch
    Dim iSeriesID As Long       'Input series ID code
    Dim calctype As Long        'for preserving current Excel calc mode

    '--- RSXRT return error codes ---
    '    0        SUCCESS -- no error conditions found
    '   -1        password/installation error. RSX output not valid
    '10011        dRSXout not declared using ByRef
    '10014        length parameter below 2
    '10015        iSeriesID not declared using ByRef
    '10016        length parameter above 500
    '10018        Cannot deallocate DLL RAM when SeriesID=0

    iSize = 100           ' length of input array
    dLength = 10          ' RSX smoothness factor
    iSeriesID = 0         ' MUST initialize series identification to zero
    iDestroy = 0          ' MUST clear "deallocate DLL RAM" flag

    'disable automatic calculation
    calctype = Application.Calculation
    Application.Calculation = xlManual

    For k = 1 To iSize
        iResult = RSXRT(Cells(k + 1, 1), dLength, dRSXout, iDestroy, iSeriesID)
        If (iResult <> 0) Then
            ' Post Error Message and HALT
            Call Error_handler(iResult, calctype)
        Else
            Cells(1 + k, 4).FormulaR1C1 = dRSXout
        End If
    Next k

    'deallocate DLL RAM. Check for errors.
    'iSeriesId should contain a non-zero identification value

    iDestroy = 1
    iResult = RSXRT(0, 0, 0, iDestroy, iSeriesID)
    If (iResult <> 0) Then
        ' Post Error Message and HALT
        Call Error_handler(iResult, calctype)
    End If

    're-enable automatic calculation
    Application.Calculation = xlAutomatic
    End Sub
```

```
' The following subroutine is a simple way to handle run-time errors that may occur
' It's good practice to handle each error type mentioned in the user manual.

Private Sub Error_handler(ByVal error_code As Long, ByVal calctype As Long)
    Dim result As Long
    result = MsgBox("Error number " & Str(error_code) &
                    " was returned by RSX.", , "RSX Error")
    Application.Calculation = calctype
    End   ' this END command will halt execution of the VBA code.
End Sub
```

IF YOU FIND A BUG . . . YOU WIN

If you discover a legitimate bug in any of our preprocessing tools, please let us know! We will try to verify it on the spot. If you are the first to report it to us, you will receive the following two coupons redeemable toward your acquisition of any of our preprocessing tools:

- a $50 discount coupon
- a free upgrade coupon

You may collect as many coupons as you can.
You may apply more than one discount coupon toward the purchase of your next tool.

$$$ Anti-Piracy Reward Policy $$$

Jurik tools are world renown for excellence and value. We manage to keep costs down with large sales volume, maintained in part by protecting our copyrights with the following anti-piracy policy…

1. We have on permanent retainer one of the best intellectual property law firms in the U.S.
2. We do not perform cost-benefit analysis when it comes to litigation. We prosecute all offenders.
3. We register portions of our software with the U.S. Copyright office, entitling us to demand the offender compensate Jurik Research for all legal costs, which is typically over $10,000 per lawsuit.
4. ***We offer up to $5000 reward for information leading to the prosecution of any offender(s).***

Risk & Liability

Hypothetical or simulated performance results have certain inherent limitations. Simulated performance is subject to the fact that they are designed with the benefit of hindsight.

We must also state here that, due to the frequently unpredictable nature of the marketplace, past performance of any trading system is never a guarantee of future performance. In addition, all trading systems have risk and commodities trading leverages that risk. We advise you never to place at risk more than you can afford to lose. It's only common sense.

The user is advised to test the software thoroughly before relying upon it. The user agrees to assume the entire risk of using the software. In no event shall JRC be responsible for any special, consequential, actual or other damages, regardless of type, and any lost profit resulting from the use of this software.