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.

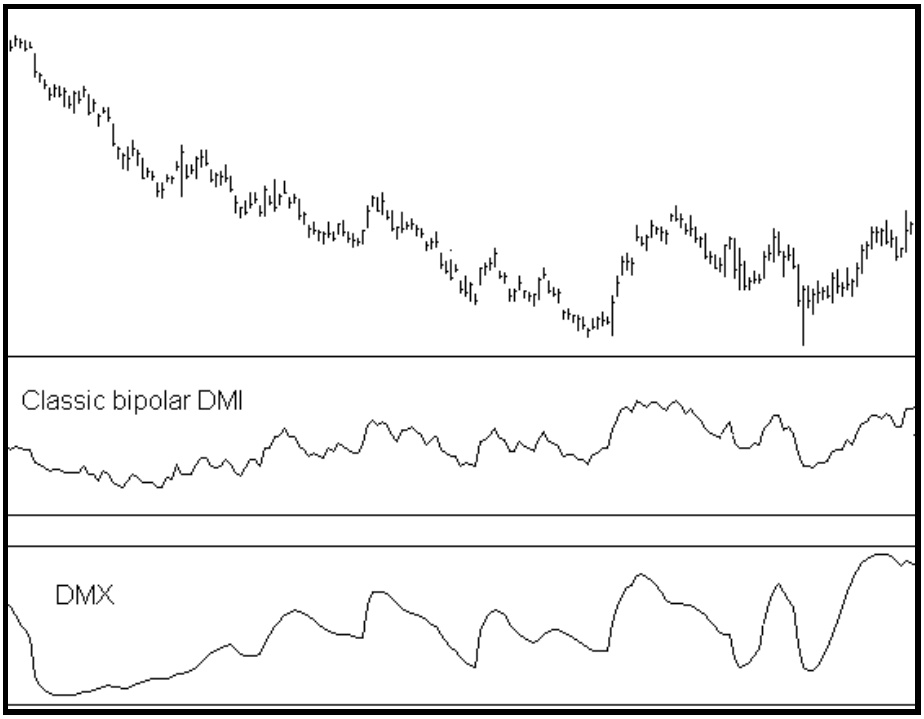
NO WAIVER -- The failure of either party to enforce any rights granted hereunder shall not be deemed a waiver by that party as to subsequent enforcement of rights in the event of future breaches.

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

DMX

Directional Movement Index
DLL module
for Windows® Application Developers



USER'S GUIDE

Requirements

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

 PO 460669, Aurora, CO 80046 303-680-4409 fax: 303-680-4605

Installing the 32 bit DLL module

- There is no separate installation procedure for DMX. If you just acquired a user license for JMA, then DMX is created when you install JMA. If you had installed JMA a long time ago, then you obtain DMX by downloading it from the "Freebie" section of Jurik Research's website.
- To operate DMX as described on the following pages, you must have JMA already installed.
- 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 JMA, DMX 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 DMX on additional computers, you will need additional passwords for JMA. For new or replacement passwords, call 323-258-4860

ABOUT DATA VALIDITY

And what to do when DMX encounters an error

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

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

Why Use DMX ?

Smoother and more responsive than DMI+ , DMI- and ADX !!

Brief Description

DMX (Directional Movement Index) is a super-smooth version of the technical indicator DMI, while retaining very fast response speed. Low-lag super-smoothing means the DMX signal has less noise (yielding fewer false alarms) and less delay than DMI. Jurik's DMX is composed of two basic functions: DMX+ and DMX-, and they are superior to the classic DMI+ and DMI-. And since ADX is slower than DMI, Jurik's DMX is superior to ADX as well.

Background

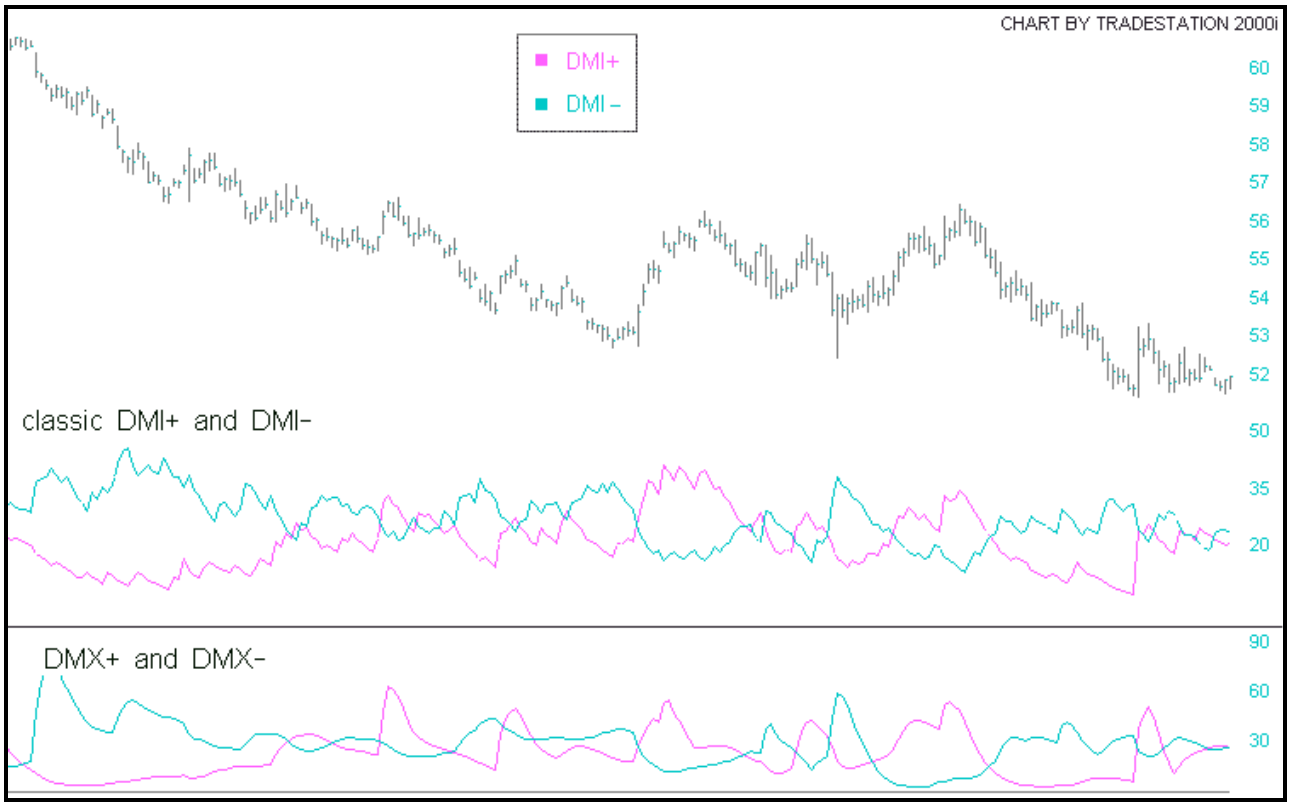
The classic DMI indicator compares upward price action to downward price action and represents their "difference" in a chart scaled from 0 to 100. A strong trend (in either direction) will produce a DMI value close to 100 and the absence of any clear trend will produce a DMI value close to 0.

The speed and clarity of the DMI signal depends on the speed and clarity of the two component signals: DMI+ which measures upward motion, and DMI- which measures downward motion. If these two signals are noisy, then DMI will also be noisy, which renders the signal unreliable for market analysis.

The source of the noise problem is that DMI+ and DMI- use the standard exponential moving average (EMA) for smoothing. Unfortunately, the EMA is a poor noise filter. In contrast, JMA is a vastly superior filter and DMX is simply DMI analysis modified to use JMA instead of EMA for smoothing purposes.

The improvement gained by using JMA is amazing.

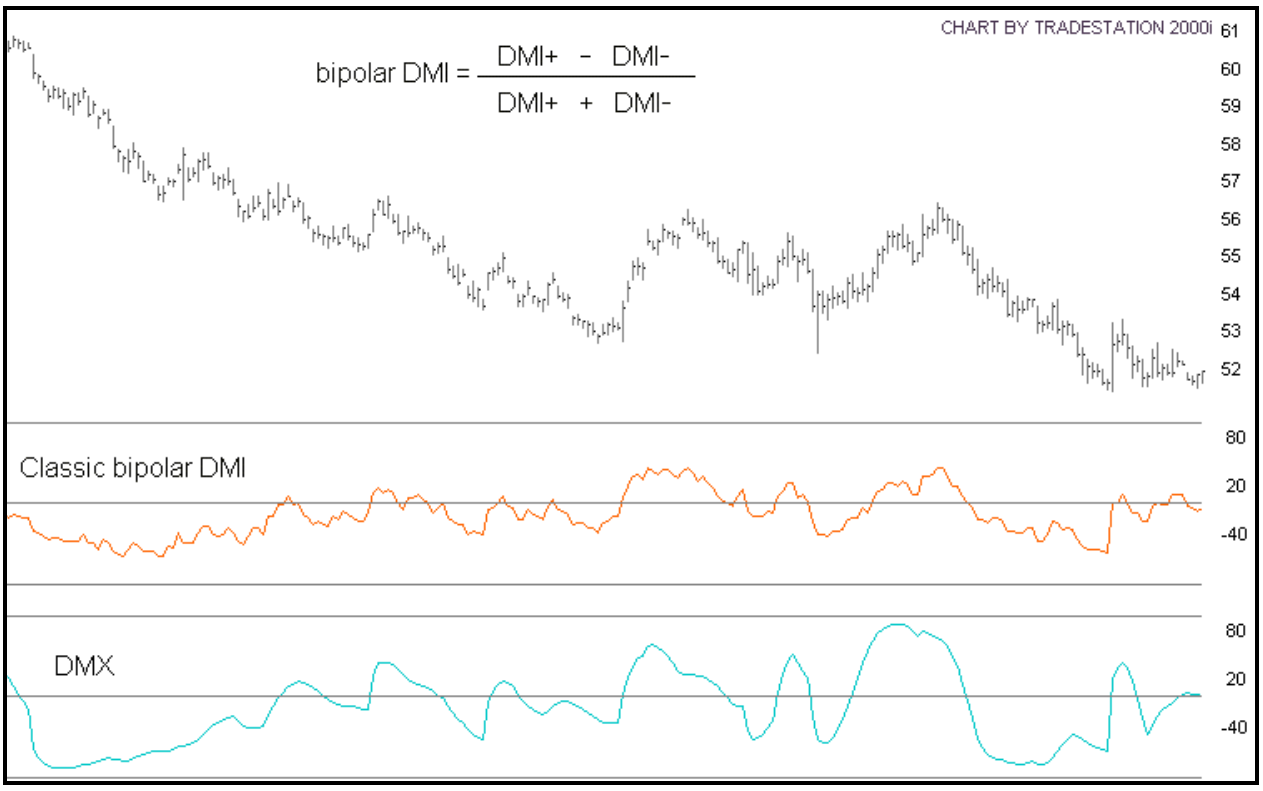
The chart below compares the classic pair of DMI+ and DMI- signals against the pair of DMX+ and DMX- signals. Note how DMX+ and DMX- reduce noise yet are just as timely at indicating market reversals.



The next chart compares DMX to a bipolar form of DMI. (Bipolar means the signal can have negative as well as positive values. Values are negative when market trend is downward.)

Here too, DMX is just as timely as DMI regarding market reversals, but virtually noise-free.

DMX has only one adjustable input parameter: LENGTH. Length controls DMX smoothness. Larger values force DMX to consider more historical points, making DMX run slower and smoother.



Coding Applications

DMX is a function that can produce three different time-series results simultaneously: DMX bipolar, DMX+ and DMX-. **DMX bipolar** is an ultra-smooth version of the classic DMI with the added capability to be negative in value when the market is trending down. **DMX+** and **DMX-** are the ultra-smooth version of DMI+ and DMI- respectively.

DMX comes in two versions, BATCH mode and REAL-TIME mode.

The **BATCH MODE** version accepts a entire arrays of input data and returns results into other arrays of equal length. This method requires the user provide the DLL function with pointers to various input and output arrays. This version is ideal when an entire array is available for processing, since it requires only one call to **DMX**.

The **REAL TIME** version accepts one period or bar of time-series data and returns results for that time bar. The process is repeated for each successive value in some arbitrary time series. This approach is ideal for processing real time data, whereby the user wants an instant **DMX** update as each new data value arrives.

The following pages cover the following applications of **DMX**:

- 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 preferring 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 DMX toolset for batch mode

If you recently purchased the Jurik tool **JMA**, then when you install **JMA**, the **DMX** function will be in the same installed DLL file as **JMA**. (**JRS_32.DLL**). On the other hand, if you purchased **JMA** a while ago and recently downloaded from the Jurik Research web site the DLL file containing free tools, (**JRS_FREE.DLL**), then you can code your software to find **DMX** in that DLL file. Either way, …

***DMX** works only when the **JMA** toolset is installed on the same computer.*

DMX can return all three time series simultaneously (DMX bipolar, DMX+ and DMX-). **DMX** achieves this by writing into arrays pointed to by the function's calling parameters. The calling parameters are specified below. If the user sets an output array pointer to NULL, then **DMX** will not calculate that specific time series result.

```
extern _declspec(dllimport) int WINAPI DMX(
     double *pdInHIGH, double *pdInLOW, double *pdInCLOSE,
     double *pdOutBipolar, double *pdOutPlus, double *pdOutMinus,
     double dLength, INT iSize );
```

PARAMETERS

pdInHIGH	A pointer to an array of doubles that contain the time series of market bar HIGH prices.
pdInLOW	A pointer to an array of doubles that contain the time series of market bar LOW prices.
pdInCLOSE	A pointer to an array of doubles that contain the time series of market bar CLOSE prices.
pdOutBipolar	A pointer to an array of doubles that the function will write its DMX bipolar results to.
pdOutPlus	A pointer to an array of doubles that the function will write its DMX+ results to.
pdOutMinus	A pointer to an array of doubles that the function will write its DMX - results to.
dLength	A double floating point that specifies smoothness of **DMX**.
iSize	A 32 bit signed integer equal to the number of doubles in each data array.

NOTES

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

iSize must be no less than 41, as the first 40 result values are forced to zero.

dLength may range from 1 to 500. Typical values range from 10 – 30.

If any output pointer is NULL, then **DMX** will not create values for the corresponding time series.

In your C code, you should declare DMX 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 (or JRS_FREE.LIB), which we provide.

RETURN VALUES

The DMX functions return an integer to indicate success or an error as follows:

```
       0     SUCCESS -- NO ERROR CONDITIONS
      -1     JMA PASSWORD/INSTALLATION ERROR
   10120     DMX REQUIRES AT LEAST 32 DATA POINTS
   10121     DMX LENGTH MUST BE BETWEEN 1 AND 500 INCLUSIVE
   10122     DMX OUT OF MEMORY
   10123     DMX MUST HAVE AT LEAST ONE VALID OUTPUT POINTER
```

C PROGRAMMING EXAMPLE for batch mode

```
iSize = 2500;
dLength = 20;

/* Input arrays */

pdInHigh     = (double *) GlobalAllocPtr( GHND, (DWORD) sizeof(double) * iSize);
pdInLow      = (double *) GlobalAllocPtr( GHND, (DWORD) sizeof(double) * iSize);
pdInClose    = (double *) GlobalAllocPtr( GHND, (DWORD) sizeof(double) * iSize);

/* Output arrays */

pdOutPlus    = (double *) GlobalAllocPtr( GHND, (DWORD) sizeof(double) * iSize);
pdOutMinus   = (double *) GlobalAllocPtr( GHND, (DWORD) sizeof(double) * iSize);
pdOutBipolar = (double *) GlobalAllocPtr( GHND, (DWORD) sizeof(double) * iSize);

/* At this location, check that memory was actually allocated, and put your time
     series data (HIGH, LOW, CLOSE) into the three input arrays. */

error_code = DMX(pdInHigh, pdInLow, pdInClose, pdOutBipolar, pdOutPlus,
pdOutMinus, dLength, iSize);

/* Check error_code */
```

You may elect to receive 1, 2 or 3 output arrays of data. To specify an output array that you do not want, simply replace the corresponding output pointer variable with “NULL” in the function call. For example, to avoid receiving OutPlus and OutMinus array data, replace the corresponding parameters with “NULL”, as in the following call …

```
error_code =
    DMX(pdInHigh, pdInLow, pdInClose, pdOutBipolar, NULL, NULL, dLength, iSize);
```

C Programming the 32 bit DMX toolset for real-time

If you recently purchased the Jurik tool **JMA**, then when you install **JMA**, the **DMXRT** function will be in the same installed DLL file as **JMA**. (**JRS_32.DLL**). On the other hand, if you purchased **JMA** a while ago and recently downloaded from the Jurik Research web site the DLL file containing free tools, (**JRS_FREE.DLL**), then you can code your software to find **DMXRT** in that DLL file. Either way, …

***DMXRT** works only when the **JMA** toolset is installed on the same computer.*

DMXRT can return all three time series simultaneously (DMX bipolar, DMX+ and DMX-). **DMX** achieves this by writing into arrays pointed to by the function's calling parameters. The calling parameters are specified below. If the user sets an output array pointer to NULL, then **DMX** will not return that specific time series result.

```
extern _declspec(dllimport) int WINAPI DMXRT(
     double dHIGH, double dLOW, double dCLOSE,
     double *pdOutBipolar, double *pdOutPlus, double *pdOutMinus,
     double dLength, int iDestroy, int *piSeriesID);
```

PARAMETERS

dHIGH	a double precision floating point number equal to the market HIGH value.
dLOW	a double precision floating point number equal to the market LOW value.
dCLOSE	a double precision floating point number equal to the market CLOSE value.
pdOutBipolar	a pointer to the memory location of a double which will receive the value of DMX bipolar
pdOutPlus	a pointer to the memory location of a double which will receive the value of DMX+
pdOutMinus	a pointer to the memory location of a double which will receive the value of DMX -
dLength	a double floating point that specifies smoothness of **DMXRT**.
iDestroy:	a 32 bit signed integer, with a value = 0 or 1. When value = 1, the RAM in the DLL used for a particular time series is released. The desired time series is designated by **piSeriesID**. (see next parameter) This event does not release the memory containing the output of **DMXRT,** e.g., the memory pointed to by pdOutPlus, etc. 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. **DMXRT** will store a unique identification number of the series into that integer (iSeriesID) pointed to by pointer piSeriesID.

NOTES

dLength may range from 1 to 500. Typical values range from 10 – 30.

Output will be zero for the first 40 times **DMXRT** is called.

In your C code, you should declare **DMXRT** 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** (or JRS_Free.lib), which we provide.

RETURN VALUES **DMXRT** returns an integer, which will indicate success or an error:

```
     0      SUCCESS -- NO ERROR CONDITIONS
    -1      JMA PASSWORD/INSTALLATION ERROR
 10121      DMX LENGTH MUST BE BETWEEN 1 AND 500 INCLUSIVE
 10122      DMX OUT OF MEMORY
 10123      DMX MUST HAVE AT LEAST ONE VALID OUTPUT POINTER
 10124      DMX ISERIESID=0 AND IDESTROY=1. CANNOT DEALLOCATE DLL RAM WITHOUT VALID SERIESID
 10125      ADDRESS OF ISERIESID CANNOT BE 0
```

C PROGRAMMING EXAMPLE for real-time mode

```
// declare variables
double *pdHigh, *pdLow, *pdCLose                    // input arrays
double *pdBipolar, *pdPlus, *pdMinus                // output arrays
double dLength                                      // control parameter
int    iSeriesID, *piSeriesID, iErr, i ;

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

// assume you want this DMX parameter value
dLength = 20 ;

// allocate RAM for input and output arrays. Assume array size is 100
pdHigh    = (double *) GlobalAllocPtr(GHND, (DWORD) sizeof(double) * 100) ;
pdLow     = (double *) GlobalAllocPtr(GHND, (DWORD) sizeof(double) * 100) ;
pdCLose   = (double *) GlobalAllocPtr(GHND, (DWORD) sizeof(double) * 100) ;
pdBipolar = (double *) GlobalAllocPtr(GHND, (DWORD) sizeof(double) * 100) ;
pdPlus    = (double *) GlobalAllocPtr(GHND, (DWORD) sizeof(double) * 100) ;
pdMinus   = (double *) GlobalAllocPtr(GHND, (DWORD) 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 DMX on each element, and store results
for(i=0;i<100;i++)
   {
   iErr = DMXRT( *(pdHigh+i), *(pdLow+i), *(pdClose+i), (pdBipolar+i),
              (pdPlus+i), (pdMinus+i), dLength, 0, piSeriesID) ;
   if(iErr != 0)
        YourErrHandlerFunc() ;
   }

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

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

// do something with data and deallocate RAM at pdHigh, pdLow, etc.
```

You may elect to receive 1, 2 or 3 output elements of data per function call. To specify an output element that you do not want, simply replace the corresponding output pointer variable with “NULL” in the function call. For example, to avoid receiving DMX_Plus and DMX_Minus data, replace the corresponding parameters with “NULL”, as in the following call …

```
iErr = DMXRT( *(pdHigh+i), *(pdLow+i), *(pdClose+i), (pdBipolar+i),
              NULL, NULL, dLength, 0, piSeriesID) ;
```

Visual Basic example of DMX in batch mode

INTRODUCTION

In your Jurik Research DLL installation directory (e.g., C:\JRS_DLL) the workbook JMA_DMX_DLL.XLS contains a programming example using Excel's VBA to call function **DMX**. (If you obtained DMX by downloading from the Jurik Research 'Free Stuff" web page, then the workbook filename is DMX_DLL.XLS). The workbook includes a worksheet where you can run the macro **DMX_Test** to run **DMX** in batch mode.

In this example, run the VBA macro called "**DMX_Test**". The macro gets data from columns 2-4 and sends it to the **DMX** batch mode function in the DLL. The three output arrays produced by **DMX** (bipolar, plus, minus) are then written back onto columns 7-9 of the worksheet.

VBA MACRO DESCRIPTION

The macro **DMX_TEST** calls the function **DMX**, which is declared as shown below. Note that the input and output arrays (daHigh, daLow, daClose, daOutX, daOutP, daOutM) are called by reference using "ByRef". This enables the calling statement to send to **DMX** a pointer to the first element of each data array.

```
    Declare Function DMX Lib "JRS_32.dll" ( _
                                ByRef daHigh As Double, _
                                ByRef daLow As Double, _
                                ByRef daClose As Double, _
                                ByRef daOutX As Double, _
                                ByRef daOutP As Double, _
                                ByRef daOutM As Double, _
                                ByVal dLength As Double, _
                                ByVal iArraySize As Long) As Long
```

You may elect to receive 1, 2 or 3 output arrays of data. To specify an output array that you do not want, find the corresponding line in the function declaration statement, change "ByRef" to "ByVal", change "Double" to "Long" and set the corresponding parameter value to zero in the function call. For example, to avoid receiving daOutP and daOutM array data, replace the relevent declaration lines with the following …

```
                                ByVal daOutP As Long, _
                                ByVal daOutM As Long, _
```

… and call **DMX** with the corresponding parameter values set to zero, as follows …

```
iResult =
   DMX(HighData(1), LowData(1), CloseData(1), OutXData(1), 0, 0, dLength, iArraySize)
```

The VBA subroutine **DMX_Test** is shown on the next page. This code will read data from columns 2-4 of the active worksheet, call the DLL function **DMX**, and output its results back to the worksheet.

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

```
Sub DMX_Test()

    Dim k As Long                       'iteration variable
    Dim j As Long                       'iteration variable
    Dim iArraySize As Long              'size of data array
    Dim iResult As Long                 'returned error code
    Dim HighData(1 To 400) As Double    'input array
    Dim LowData(1 To 400) As Double     'input array
    Dim CloseData(1 To 400) As Double   'input array
    Dim OutPData(1 To 400) As Double    'DMX plus output array
    Dim OutMData(1 To 400) As Double    'DMX minus output array
    Dim OutXData(1 To 400) As Double    'DMX output array
    Dim dLength As Double               'DMX speed
    Dim calctype As Long

    calctype = Application.Calculation
    Application.Calculation = xlManual

    iArraySize = 400    'size of input series
    dLength = 20        'DMX speed

    ' Read Data from spreadsheet into array
    ' Input data are in columns 2, 3, and 4

    For k = 1 To iArraySize
        j = k + 2
        HighData(k) = Cells(j, 2)
        LowData(k) = Cells(j, 3)
        CloseData(k) = Cells(j, 4)
    Next k

    '--- DMX return error codes ---
    '    0      SUCCESS -- No error conditions
    '   -1      JMA password/installation error
    '10120      DMX requires at least 32 data points
    '10121      DMX Length must be between 1 and 400 inclusive
    '10122      DMX Out of Memory
    '10123      DMX must have at least one valid output pointer

    ' Call DMX using pointers to first elements of arrays

    iResult = DMX(HighData(1), LowData(1), CloseData(1), OutXData(1), OutPData(1), _
    OutMData(1), dLength, iArraySize)

    If (iResult <> 0) Then
        ' Post Error Message and HALT
        Call DMX_Error_handler(iResult, calctype)
    Else
        ' Show results in columns 7,8, and 9 on spreadsheet
        For k = 1 To iArraySize
            j = k + 2
            Cells(j, 7).FormulaR1C1 = OutPData(k)
            Cells(j, 8).FormulaR1C1 = OutMData(k)
            Cells(j, 9).FormulaR1C1 = OutXData(k)
        Next k
    End If

    Application.Calculation = calctype

    End Sub
```

Visual Basic example of DMX in real time mode

INTRODUCTION

In your Jurik Research DLL installation directory (e.g., C:\JRS_DLL) the workbook JMA_DMX_DLL.XLS contains a programming example using Excel's VBA to call function **DMXRT**. (If you obtained DMX by downloading from the Jurik Research 'Free Stuff" web page, then the workbook filename is DMX_DLL.XLS). The workbook includes a worksheet where you can run the macro **DMXRT_Test** to run **DMXRT** in real-time mode.

In this example, run the VBA macro called "**DMXRT_Test**". The macro reads one row of three elements at a time from columns 2-4, sequentially feeding each one through the real time version of **DMX** and places the results sequentially into columns 11-13.

VBA MACRO DESCRIPTION

The function **DMXRT** is declared as shown below. Note that the output and series identification variables (dOutX, dOutP, dOutM and iSeriesID) are called by reference using "ByRef". The user initializes the series identification variable (iSeriesID) to zero and during the first call to **DMXRT**, the function will replace zero with an integer that uniquely identifies the time series set (High, Low, Close). This way, when you have multiple time series running in parallel, the series identification numbers will tell **DMXRT** to which time series set the new input data points are to be assigned.

```
Declare Function DMXRT Lib "JRS_32.dll" ( _
                           ByVal dHigh As Double, _
                           ByVal dLow As Double, _
                           ByVal dClose As Double, _
                           ByRef dOutX As Double, _
                           ByRef dOutP As Double, _
                           ByRef dOutM As Double, _
                           ByVal dLength As Double, _
                           ByVal iDestroy As Long, _
                           ByRef iSeriesID As Long) As Long
```

You may elect to receive 1, 2 or 3 output arrays of data. To specify an output array that you do not want, find the corresponding line in the function declaration statement, change "ByRef" to "ByVal", change "Double" to "Long" and set the corresponding parameter value to zero in the function call. For example, to avoid receiving dOutP and dOutM array data, replace the relevent declaration lines with the following …

```
                           ByVal dOutP As Long, _
                           ByVal dOutM As Long, _
```

… and call **DMXRT** with the corresponding parameter values set to zero, as follows …

```
iResult =
   DMXRT(Cells(j, 2), Cells(j, 3), Cells(j, 4), dOutX, 0, 0, dLength, 0, iSeriesID)
```

The VBA subroutine **DMXRT_Test** is shown on the next page. This code reads data from column 2-4 of the active worksheet, one row of three elements at a time, each time calling the DLL function **DMXRT**, and outputting the result back to the worksheet.

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

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

```
Sub DMXRT_test()

    Dim dHigh As Double         'Market High
    Dim dLow As Double          'Market Low
    Dim dClose As Double        'Market Close
    Dim dOutX As Double         'DMX Bipolar output
    Dim dOutP As Double         'DMX Plus output
    Dim dOutM As Double         'DMX Minus output
    Dim dLength As Double       'DMX speed
    Dim iSeriesID As Long       'Input series ID code
    Dim iResult As Long         'returned error code
    Dim iArraySize As Long      'length of data array
    Dim k As Long               'iteration variable
    Dim calctype As Long        'store current calculation type

    '--- DMXRT return error codes ---
    '    0   SUCCESS -- No error conditions
    '   -1   JMA password/installation error
    '10121   DMX Length must be between 1 and 500 inclusive
    '10122   DMX Out of Memory
    '10123   DMX must have at least one valid output pointer
    '10124   DMX Cannot deallocate DLL RAM without valid SeriesID
    '10125   address of iSeriesID cannot be 0

    iArraySize = 400      ' length of input array
    dLength = 20          ' DMX smoothness factor
    iSeriesID = 0         ' MUST initialize series identification to zero

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

    For k = 1 To iArraySize
        j = k + 2
        iResult = DMXRT(Cells(j, 2), Cells(j, 3), Cells(j, 4), dOutX, dOutP, dOutM, _
        dLength, 0, iSeriesID)

        If (iResult <> 0) Then
            ' Post Error Message and HALT
            Call DMX_Error_handler(iResult, calctype)
        Else
            Cells(j, 11).FormulaR1C1 = dOutP
            Cells(j, 12).FormulaR1C1 = dOutM
            Cells(j, 13).FormulaR1C1 = dOutX
        End If
    Next k

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

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

    'restore calculation type
    Application.Calculation = calctype

    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.