# *Real-Time Data Exchange*

*WHITE PAPER: SPRY012*

*Deborah Keil*
*Technical Lead, RTDX Project*
*Software Development Systems*

*Digital Signal Processing Solutions*
*February 1998*

TEXAS INSTRUMENTS

## IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

## TRADEMARKS

TI and RTDX are trademarks of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

**CONTACT INFORMATION**

| | |
|---|---|
| US TMS320 HOTLINE | (281) 274-2320 |
| US TMS320 FAX | (281) 274-2324 |
| US TMS320 BBS | (281) 274-2323 |
| US TMS320 email | dsph@ti.com |

# Contents

# Figures

# Real-Time Data Exchange

## Abstract

In the past, DSP application designers have gathered data by stopping a target application at designated breakpoints to read registers and other data storage locations. This practice is not only cumbersome, it can provide misleading data because it yields only a snapshot obtained by suddenly stopping a high-speed application — a readout that may not present an accurate view of the system's continuous operation.

This paper presents the Texas Instruments (TI™) digital signal processing (DSP) analysis technology, Real-Time Data Exchange (RTDX™). RTDX offers developers continuous bi-directional data exchange in real time with minimal perturbation on the application. Because RTDX uses TI's universal JTAG data path and TI's debugger, it can be supported on every TI processor without assuming any particular I/O peripherals and can also use other data interfaces instead of or in addition to JTAG.

RTDX displays data using your favorite OLE-enabled visualization package, is easy to program on both target and host, and is provided at no additional cost, bringing substantial value to TI's DSP solutions.

## Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.

# Introduction

The best analysis techniques for DSP systems are similar to those for medical diagnosis in that they require accurate, real-world information for proper results. For example, doctors put cardiac patients through stress tests that continuously monitor vital signs in changing conditions rather than record information only at intervals. This method provides real-time diagnostics: a stress test carefully analyzes patient health by monitoring changes and their causes *as they occur in the body*.

This same continuous monitoring can be applied to the world of DSP through Texas Instruments' new DSP analysis technology, Real-Time Data Exchange (RTDX).

RTDX enables developers to transmit and receive data between a host computer and a target system without stopping their applications. Output data can be directed to customized and third-party visualization tools via the industry-standard object linking and embedding (OLE) application program interface (API). RTDX speeds and enhances the accuracy of DSP application debugging, saving developers time in preparing their products for the market.

RTDX is provided at no additional cost, bringing substantial value to TI's DSP solutions.

## Uses for RTDX

RTDX capability enables easy analysis for many current and emerging DSP systems. Wireless telecommunication designers can capture the output of their vocoder algorithms to check the implementations of speech applications.

Embedded control systems also benefit. For example, hard disk drive applications can be tested without improper signals to the servomotor crashing the drive. Engine control designers can analyze changing conditions such as heat and environmental conditions while the control application is running. In all uses of RTDX, designers can view performance through the most meaningful visualization tools for these applications.

# RTDX Data Flow

The last two letters of the acronym RTDX stand for “Data Exchange” — precisely what this technology provides. Data can be sent from the target to the host and from the host to the target.

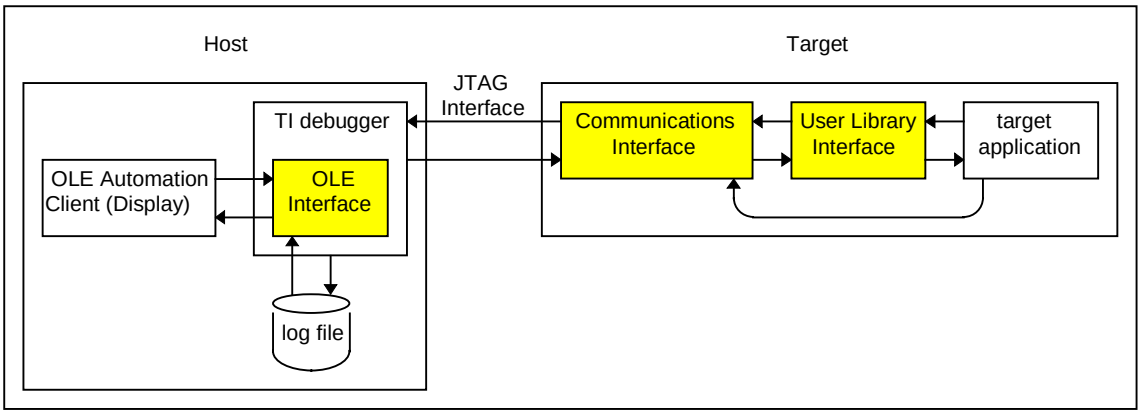
Baseline support for data exchange is supplied by the emulation logic included on every TI DSP, so data can be transferred using the same simple JTAG debug connection used for all other debug/control operations. This fact means RTDX can be supported on every TI processor without assuming any particular I/O peripherals. However, RTDX technology can also take advantage of other data interfaces either instead of or in addition to JTAG. Customers are free to use data paths that provide higher bandwidths or more communication security in end-user products.

In addition to providing a universal connection, the JTAG interface and emulation logic also serve to minimize the intrusiveness of the RTDX capability. Advanced emulation allows data to be transferred to the host as a background activity with minimal perturbation of the processor. In processors with advanced emulation, each data word is transferred directly from memory. The hardware may borrow a processor cycle for each word or do direct memory access depending on the processor implementation.

## Target to Host

The TI debugger controls the flow of data between the host and target (see Figure 1). Data flows from the target application through the User Library Interface and the Communications Interface to the TI debugger running on the host. The target application calls routines in the User Library Interface that buffer the data and pass it to the Communications Interface. This in turn sends the data to the debugger by way of a JTAG interface. The debugger records the data in a log file on the host.

*Figure 1. RTDX Data Flow*




On the host, the TI debugger supports OLE automation. Any host application that is an OLE Automation Client (those written in Visual Basic or Visual C++, for example) can access data in an RTDX log file or send data to the target via the OLE interface.

## Host to Target

Data flows from the OLE Automation Client to the TI debugger. The debugger buffers the data on the host. When the debugger receives a read request from the target application, the data is sent to the Communications Interface on the target via a JTAG interface. The Communications Interface passes the data to the User Library Interface, which then delivers it to the target application.

## Putting RTDX into Action

A simple example illustrates the ease of using RTDX. A target application will use RTDX to send 100 consecutive integers to the host. The data will be displayed in a Microsoft Excel spreadsheet.

## Preparing the Target Application

Normally, RTDX User Library Interface calls are inserted into a working target application. The target application in this example is a loop that generates an array of data. RTDX calls were inserted to open and enable an output channel and log the generated data to that channel. RTDX sends the data to the host where it is recorded in a log file. The C code on the target is shown in Figure 2. The RTDX commands are shown in boldface type.

In a real application, RTDX_Data_Write would be called to log data at the rate at which the real-time application generated it. This toy example generates data at an extremely high rate limited only by CPU speed. In the example, we are using the return value of RTDX_Data_Write to determine if the data was successfully logged. The surrounding while loop does a busy wait until the data can be accepted. The data rate in a real application would be determined by the function it performed; the while loop would not be used.

*Figure 2. Target Application Code Written in C*

```
/*********************************************************************
* - Transmits 100 integers, 10 at a time.
* - Shows Data transmission from target to host.
*********************************************************************/

#include <RTDX_usr.h>               /* RTDX User Library Interface   */
#include <stdio.h>                  /* printf                        */

#define DATA_MEMBERS    10          /* number of elements in array   */

RTDX_output_channel ochan;          /* globally declare RTDX channel */

static void Generate_Application_Data( int *dp );

void main( void )
{
  int data[DATA_MEMBERS];              /* data to send to host       */
  int i;

  RTDX_Enable_Data_Output(&ochan);

  for (i = 0; i < 10; i += 1)          /* for 10 messages...         */
  {
    Generate_Application_Data(data);   /* generate data              */
    while (                           /* limit TOY EXAMPLE data rate */
       !RTDX_Data_Write( &ochan, data, sizeof(data) )   /* send data  */
    );
  }

  RTDX_Disable_Data_Output(&ochan);

  printf("\n\nTest Completed");

} /* main */

/* Our toy application generates arrays of integers.                 */
static void Generate_Application_Data( int dp[] )
{
  int i;
  for (i = 0; i < DATA_MEMBERS; i++)   /* for RTDX message size...   */
    dp[i] = i;                         /* fill array with data       */
}
```

## Preparing the Host Application

Host-side RTDX declarations and calls to the OLE Interface can be inserted to obtain the data from the target. Refer to the Microsoft Excel Visual Basic code in Figure 3. This code is executed within an Excel Workspace. The RTDX commands are again shown in boldface type.

*Figure 3. Visual Basic Module Executed Within an Excel Workspace on the Host*

```
###############################################################################
' Copyright (c) 1997 Texas Instruments Incorporated
' A module to read data from a single channel into an Excel spreadsheet
' that is automatically cleared & updated with new data.
###############################################################################

' RTDX OLE API Status Return codes
Const SUCCESS = &h0
Const FAIL = &h80004005
Const ENoDataAvailable = &h8003001e
Const EEndOfLogFile = &h80030002

Const members = 10                          ' Total number of integers in array

Option Explicit
Sub read_channel()
  Dim rtdx As Object                        ' object variable
  Dim row, col As Integer                   ' temporary worksheet indices
  Dim status As Long                        ' variable for status of RTDX read
  Dim data As Long                          ' variable for data from RTDX read

  Worksheets("Data").Range("A1:IV16384").Clear  ' Clear old data

  Set rtdx = GetObject("", "Debugger.RTDX")     ' Get Debugger class object
            ' variable RTDX becomes an instantiation of the OLE interface
  status = rtdx.Open("ochan", "R")  ' Open channel with name "ochan" for reading

  row = 1                                       ' Start data display here
  col = 1

  Do
    status = rtdx.ReadI4(data)               ' Read value from data channel
    Select Case status
      Case Is = SUCCESS                         ' if ReadI4 was successful...
        Worksheets("Data").Cells(row, col) = _
           data                                 ' Copy data read into spreadsheet
        col = col + 1
        If (col > members) Then             ' if reading next column of data...
          row = row + 1                             ' move to next row
          col = 1                                   ' column 1
        End If
      Case Is = FAIL                            ' if ReadI4 failed...
        MsgBox ("Error reading data")
        Exit Do
      Case Is = ENoDataAvailable                ' if data not yet available...
        MsgBox ("Data not Available")
      Case Is = EEndOfLogFile                   ' if end of data reached...
        MsgBox ("End of Log reached")
        Exit Do
      Case Else
        MsgBox ("Unknown return code")          ' trap invalid return codes...
        Exit Do
    End Select
  Loop Until status = EEndOfLogFile

  status = rtdx.Close()                         ' Close channel

End Sub
```

## Obtaining the Data

Once the target application and host applications are prepared, they must be executed. Using the TI debugger, the target application is loaded onto the DSP and run. The data passed into RTDX_Data_Write is buffered and passed on to the Communications Interface. A scan-based emulator built into the DSP core moves the data from the DSP to the host via a JTAG interface (serial test bus). On the host, the TI debugger records the data it receives in a log file. This entire operation is done in real time.

The host application must also be executed to obtain the data from the log file. The host application can be executed simultaneously with the target application or at a later time. If executed simultaneously, the data is delivered to the host application as it becomes available. If executed after the target application, the data is immediately available from the log file.

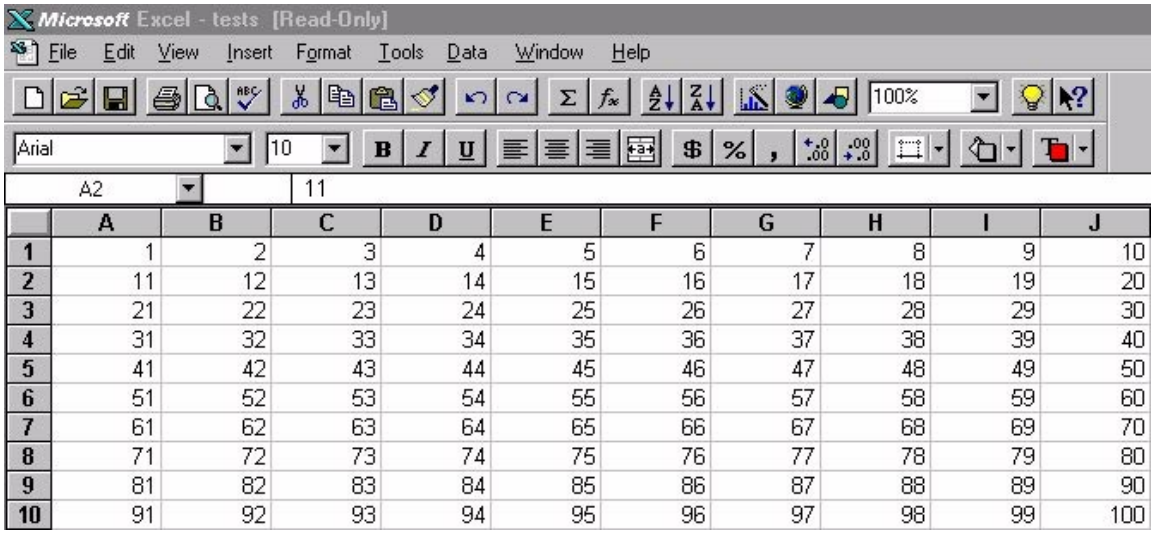
If the TI debugger is not currently running and the host application is executed on an existing log file, the TI debugger will be invoked by the host application as an OLE Automation Server. The TI debugger is the agent that allows the data to be obtained.

## Displaying the Data

The compatibility of RTDX with industry-standard APIs benefits DSP system developers by allowing them to choose the most appropriate visualization tool and focus their attention on debugging systems rather than obtaining data. Data in the log file can be accessed by any host application that is an OLE Automation client.

Commercially available application development software packages include Microsoft Excel, National Instruments' LabVIEW and Quinn-Curtis' Real-Time Graphics Tools. Alternatively, the host application can be written in Visual Basic or Visual C++. Visual Basic may be used within Microsoft Excel, as is done in this example. The result is shown in Figure 4.

*Figure 4. Data Displayed in Microsoft Excel*




| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 2 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 3 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 4 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 5 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 6 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 7 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| 8 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| 9 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 10 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |

## Summary

The ability to continuously monitor the performance of TMS320 DSP applications — as though viewing a dashboard on a car — will significantly save time for third party developers and end users.

To review its key benefits, RTDX:

- Provides continuous bi-directional data exchange without halting an application
- Functions in real time with minimal perturbation on the application
- Uses TI's universal JTAG data path and TI's debugger
- Displays data using your favorite OLE-enabled visualization package
- Is easy to program on both target and host
- Is a capability offered at no additional cost, bringing substantial value to TI’s DSP solutions