Application Report
SPRA872A – January 2003

How to Optimize Your Target Application for RTDX Throughput

Dustin Allensworth and Jason Smathers *Software Development Systems/RTDX Team*

ABSTRACT

Real-Time Data Exchange (RTDX™) is a technology developed by Texas Instruments that enables real-time bi-directional communication between a digital signal processor (DSP) or microcontroller and a host application.

This document describes how to increase the target-to-host throughput of an RTDX target application. This document does **not** describe how to increase the host-to-target throughput, although the same principles apply.

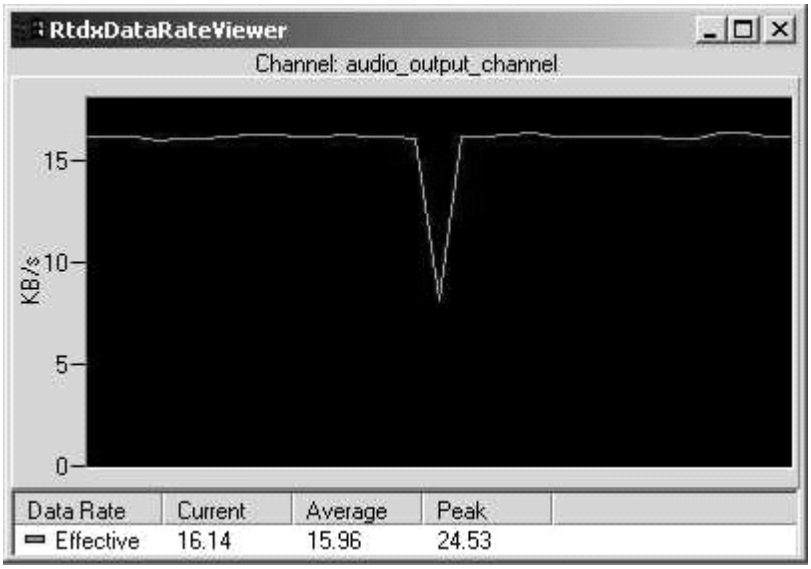
Although target-to-host data transfers occur in real-time, host-to-target transfers do not. The reason is because, in order for a host-to-target transfer to take place, the target must first request the data. And, a host application must supply the data to send. Two data transmissions occur for host-to-target instead of one like in target-to-host.

This document assumes that you have Code Composer Studio™ v2.12 or higher and the RTDX Data Rate Viewer Control. If you do not have the RTDX Data Rate Viewer Control, you can download and install it from the following link:
https://www-a.ti.com/downloads/sds_support/CCS-2.12.01-SA-to-TI-RTDXDATRAT.htm
This document also assumes that y ou are familiar with using RTDX within Code Composer Studio. If you are not familiar with RTDX, please review the RTDX tutorial.

Contents

1 Introduction to the Data Rate Viewer Control

The RTDX Data Rate Viewer Control is an ActiveX control that monitors and displays the effective target-to-host throughput of your RTDX target application. Specifically, the control displays the current, average and peak data rates for a particular target-declared output channel. The control also provides an option to display the amount of bandwidth consumed by RTDX transfer overhead. Use the control to diagnose target-to-host transfer issues and to measure data rate improvements as you optimize your target application following the RTDX performance considerations guidelines.



This control does **not** measure host-to-target throughput.

More information on using the Data Rate Viewer Control is provided with the control's online help.

2 Lab: System Requirements

Before you begin, be sure to have the following:

- Code Composer Studio (version 2.12 or higher) installed
- RTDX Data Rate Viewer Control installed and accessible
  To open the RTDX Data Rate Viewer Control, from Code Composer, select Tools→RTDX→Data Rate Viewer Control. (If the Data Rate Viewer Control sub-menu is not visible, see the Known Issues section).

3 Lab: Setup

The lab exercise is an experiment with the source code located at the end of this section.

1. Invoke Code Composer
2. Create a new project called datarate under the \myprojects directory (default location: C:\ti\myprojects).

3. Add (Ctrl+N) a new source file to your project called datarate.c and copy the source code from this document into datarate.c
4. Include the necessary linker control file (*.cmd), interrupt vector assignments (*.asm), RTDX target (rtdx*.lib) & C Runtime libraries (rts*.lib) for building the application datarate.out.

   See the RTDX tutorials or examples to identify the required files to build your RTDX target application.
5. If you are on a C54x device, set up the timer interrupt vector to call `_timer_fnc`. You can set this up by editing *intvecs.asm* and inserting the following code after the interrupt `TINT_V`:

```
.ref _timer_fnc
B        _timer_fnc
```

4 Lab: Experiment with Target-To-Host Example

This lab is an experiment with the source code located at the end of this section.

- Reduce usage of standard I/O:

  The source code contains a `fprintf()` statement that prints the data that was transmitted from the target to the host. Although this statement is helpful for validating data, it affects the throughput of the application. This experiment will show how a single `fprintf()` statement after every data transmission affects the throughput.

  1. Build and load the application onto the target.
  2. Enable RTDX.
  3. Bring up the RTDX Data Rate Viewer Control

     From the Tools menu, select RTDX.

     From the RTDX sub-menu, select Data Rate Viewer Control.
  4. Right-click on the Data Rate Viewer Control.
  5. From the context menu, select the Property Page ... menu option.
  6. From the property page, select ochan from the Channel combo box.
  7. Press the OK button to save the settings and close the property page.
  8. Run the target application and view the data rate.

     Make a note of the data rate.
  9. Edit datarate.c and assign the value **0** to symbol `FPRINTF`.
  10. Rebuild and rerun the application.

      Note the significant increase in the data rate.
- Reduce the OVERHEAD:

  There is a certain amount of overhead with each message that is transferred from the target to the host. The more data transfers (calls to `RTDX_write()`), the more accumulation of overhead involed in your target applications throughput. This experiment will show how the overhead is reduced by transmitting arrays of data instead of single integers with each message.

1. Right-click on the Data Rate Viewer Control.
2. From the context menu, select the Property Page ... menu option.
3. From the property page, ensure a check within the check boxes (☑) labeled Plot Overhead Rate and Show Overhead Rate.
4. Press the OK button to save the settings and close the property page.
5. Rerun the application.

   Notice the significant amount of overhead with sending `MAX_ELEMENTS` to the host, one integer at a time.
6. Edit datarate.c and assign the value **1** to symbol `WRITE_ARRAY`.
7. Rebuild and rerun the application.

Notice the significant decrease in overhead.

- Limit how often you mask off the RTDX interrupt

  If you are working on an interrupt driven implementation of RTDX (C6x and C55x), masking off the RTDX interrupt while doing some work within a busy loop can also affect the throughput of your target application. This experiment will insert a busy loop, which will temporarily mask off the RTDX interrupt. After the loop has completed, the RTDX interrupt will be re-enabled.

  1. Edit datarate.c and assign the value **100** to symbol `MAX_BUSYCOUNT`.
  2. Rebuild and rerun the application.

     Notice the decrease in the data rate.

- Call `RTDX_Poll()` more often

  If you are working with a polling driven implementation of RTDX such as C54x, calling `RTDX_Poll()` more often will increase the target-to-host throughput. This experiment will instrument a timer function (`timer_fnc()`) that will call `RTDX_Poll()`. You can control how often it is called by assigning a value to the symbol `TIMER_FREQ`, which sets the timer period register (`PRD`).

  Before you begin, be sure that the timer interrupt vector calls `_timer_fnc`. You can set this up by editing *intvecs.asm* and inserting the following code after the interrupt, `TINT_V:`

```
.ref _timer_fnc
B       _timer_fnc
```

  1. Edit datarate.c and assign the value **0xFFFF** to symbol `TIMER_FREQ`.
  2. Rebuild and rerun the application.

  Notice the increase in the data rate.

5 Code

```
/*****************************************************************
 * FILENAME: $RCSfile: datarate.c,v $
 * VERSION : $Revision: 1.2 $
 * DATE    : $Date: 2002/8/02 11:53:26 $
 * Copyright (c) 1997–2002 Texas Instruments Incorporated
 *
 * Target Write Example:
 * ____________________________________________________________________
 * – Uses ONE output channel.
 *
 * – This is the module to be run on the TARGET.
 * – This program is meant to be used with the RTDX Data Rate Control.
 *****************************************************************/
#include <stdio.h>                      /* fprintf(), puts()        */
#include <stdlib.h>                     /* abort()                  */
#include <rtdx.h>                       /* RTDX                     */
#include ”target.h”                     /* TARGET_INITIALIZE()      */
/* Modify WRITE_ARRAY value:
 * 0 => write one integer per message
 * 1 => write an array of size MAX_ELEMENTS for each message
 */
#define WRITE_ARRAY 0
/* Modify FPRINTF value:
 * 0 => don’t print to stdout
 * 1 => print to stdout after every message
 */
#define FPRINTF 1
/* Modify MAX_BUSYCOUNT value:
 * 0 => do not use a busy loop
 * 1 ... sizeof(int) => busy loop counter
 */
#define MAX_BUSYCOUNT 0
/* Modify TIMER_FREQ value:
 * 0x0 => do not use timer
 * 0x1 ... sizeof(int) => timer frequency
 */

#define TIMER_FREQ 0x0
#define MAX_MESSAGES 500
#define MAX_ELEMENTS 5          /* MAX_ELEMENTS only used with array */
/* Declare and initialize an output channel called ”ochan”          */
RTDX_CreateOutputChannel(ochan);
void interrupt timer_fnc()
{
#if (RTDX_POLLING_IMPLEMENTATION)
        RTDX_Poll();
#endif
}
void BusyLoop(int count)
{
        unsigned int saved_enable_state=0;
        int i;
```

```
#if (!RTDX_POLLING_IMPLEMENTATION)
#ifdef _TMS320C6X
        /* Save RTDX enable state                                   */
        saved_enable_state = IER;
        /* Mask off RTDX interrupt (RESV2, bit #3)                  */
        IER &= 0xFFFFFFF2;
#endif
#ifdef __TMS320C55X__
#define IER1 0x0045
        /* Save RTDX enable state                                   */
        saved_enable_state = (*(volatile short *) IER1);
        /* Mask off RTDX interrupt (DLOGINTE, bit #9)               */
        (*(volatile short *) IER1) &= 0xFDFF;
#endif
#endif
        for (i=0; i<count; i++) {};
#if (!RTDX_POLLING_IMPLEMENTATION)
#ifdef _TMS320C6X
        /* Restore RTDX enable state                                */
        IER = saved_enable_state;
#endif
#ifdef __TMS320C55X__
#define IER1 0x0045
        /* Restore RTDX enable state                                */
        (*(volatile short *) IER1) = saved_enable_state;
#endif
#else
        RTDX_Poll();
#endif
        return;
}
void main()
{
        unsigned int i = 0;
        #if WRITE_ARRAY
                int array[MAX_ELEMENTS];
                unsigned int x = 0, last = 0;
        #endif
        /* Target initialization for RTDX                           */
        TARGET_INITIALIZE();
        #if (RTDX_POLLING_IMPLEMENTATION)
        #ifdef _TMS320C5XX
        #if TIMER_FREQ
                #define IMR 0x0
                #define INTM 0x11
                #define TIM 0x24
                #define PRD 0x25
                /* enable timer (TINT, bit #4)                      */
                (*(volatile short *) IMR) |= 0x8;
                /* set timer frequency                              */
                (*(volatile short *) PRD) = TIMER_FREQ;
                /* enable interrupts  (INTM, bit #11)               */
                asm (”  RSBX 1, INTM”);
```

```
        #endif
        #endif
        #endif

        /* Enable the output channel, "ochan"                          */
        RTDX_enableOutput(&ochan);
        for ( i = 0; i < MAX_MESSAGES; i++ ) {
                #if WRITE_ARRAY
                        /* Build the array                             */
                        for ( x = last;
                              x < (last + MAX_ELEMENTS);
                              x++ )
                                { array[x % MAX_ELEMENTS] = x; }
                        last = x;
                #endif
                /* Send the data to the host                           */
                if (!RTDX_write(&ochan,
                        #if WRITE_ARRAY
                                array,
                                sizeof(array) ) )
                        #else
                                &i,
                                sizeof(int) ) )
                        #endif
                {
                        fprintf(stderr,
                                "\nError: RTDX_write() failed!\n");
                                abort();
                }
                /* Wait for data transfer                              */
                while ( RTDX_writing != NULL ) {
                        #if (MAX_BUSYCOUNT)
                                BusyLoop(MAX_BUSYCOUNT);
                        #endif
                        #if (RTDX_POLLING_IMPLEMENTATION)
                                RTDX_Poll();
                        #endif
                }
                #if FPRINTF
                        /* Display the message number in stdout        */
                        fprintf(stdout,
                                "Message %i was sent to the host\n",
                                i);
                #endif
        }
        /* Disable the output channel, "ochan"                         */
        RTDX_disableOutput(&ochan);
        puts("\nApplication Completed!");
}
```

6 FAQ

1. **Why is the caption “ALL is not a valid channel” displayed?**

The caption “ALL is not a valid channel” will be displayed until a program containing an RTDX output channel has been successfully loaded onto the target. For channels other than ALL, verify that the channel you are trying to monitor is declared in your RTDX target application.

2. **How do I monitor the data rate for multiple channels?**

It is possible to monitor the data rate for multiple channels by opening multiple instances of the RTDX Data Rate Control and selecting a valid output channel from the property page for each instance.

3. **Does the Data Rate Viewer Control monitor host-to-target channels?**

No, the RTDX Data Rate Viewer Control will only monitor target-to-host channels.

4. **Why does the data rate sometimes spike/drop to zero?**

Unexpected spikes in the data rate can result from large message transfers that take longer than (or a substantial portion of) the refresh interval. For example, assume the control’s refresh rate is set at 1 second and you are sending messages that take 1.5 seconds each to transfer. In this case, the control will display zero on the first refresh (after 1 second) because, the message has not yet been completely transferred. However, on the second refresh (after 2 seconds) the message has transferred completely, and the appropriate data rate will be displayed. To avoid spikes in the data rate, increase the refresh rate on the control’s property page so that the refresh interval is large compared to the time it takes to transfer one message.

5. **Why does the data rate drop to zero when I click on something in Code Composer Studio?**

Because the host-side (core) functionality of RTDX currently runs within the same thread as Code Composer Studio, some user operations (such as clicking on a menu item in Code Composer Studio) can cause RTDX to become temporarily suspended (until, for example, the user clicks off of the menu item).

6. **What is the difference between the “Effective Rate” and the “Overhead Rate”?**

The “Effective Rate” is the rate of the target application data. The “Overhead Rate”, on the other hand, is the rate of RTDX transfer overhead. In other words, it is the amount of bandwidth consumed by the RTDX transfer overhead. You can reduce this overhead by sending larger RTDX messages.

7 Known Issues

Data Rate Viewer Control does not appear in Code Composer Studio’s RTDX sub-menu.

With Code Composer Studio, version 2.1, there is a problem with registering **new** RTDX ActiveX controls into the existing Code Composer Studio RTDX group of plug-ins. When the control is registered, the Code Composer Studio component (plug-in) manager recognizes the new plug-in. However, the plug-in manager does not know how to append the new control to an existing group. Therefore, hand editing of the component manager’s database file is required. The following steps will instruct you on how to do this:

- Close Code Composer Studio.
- Register the control
  Example:

  ```
  C:\ti\plugins\rtdx>regsvr32 RtdxDataRate.ocx
  ```
- Start Code Composer Studio
- If control is still not present from RTDX sub-menu:
  1. Change directory (cd) to C:\Program Files\Common Files\Texas Instruments.
  2. Make sure there is a backup copy of Code Composer Studio_Compdb.ini.
  3. Open Code Composer Studio_Compdb.ini in an editor window.
  4. Search for entry, **RtdxDataRateViewer**.
  5. Once you have located the entry for the Data Rate Viewer Control, highlight and copy the plug-in number (**PlugIn_#**) to your clipboard.
  6. PageUp to the entry labeled, **PlugInLst_GrpName**.
  7. Append the plug-in number (PlugIn_#) from your clipboard to the end of the **PlugInLst_GrpName** field.
  8. Save file.
  9. Re-invoke Code Composer Studio.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265