LabVIEW™ DSP Test Integration Toolkit for TI DSP

The LabVIEW DSP Test Integration Toolkit for TI DSP gives you the ability to use LabVIEW and the TI Code Composer Studio™ (CCS) Integrated Development Environment (IDE) to create test systems for DSP target code.

The LabVIEW DSP Test Integration Toolkit includes VIs to automate CCS IDE and VIs that use the TI Real-Time Data Exchange™ (RTDX™) software technology to communicate with TI DSP development boards that support RTDX. Use the toolkit automation VIs to work with CCS IDE project (`.pjt`) files programmatically. Use the toolkit RTDX communication VIs and the toolkit memory VIs to programmatically exchange data with target code. The LabVIEW DSP Test Integration Toolkit also includes the LabVIEW Debugging Workbench for RTDX™ Communication, which you can use to interact with RTDX channels on development boards that support RTDX.

Refer to the *LabVIEW DSP Test Integration Toolkit Help* by selecting **Help»LabVIEW DSP Test Integration Toolkit Help** for reference information about the DSP Test Integration VIs.

Contents

Installation 2
Automating CCS IDE Functions 3
Exchanging Data with DSP Target Code 3
Writing and Reading Data from DSP Memory 4
Working with Target Code Data in LabVIEW 5
Using DSP Test Integration Advanced VIs 5

LabVIEW™, National Instruments™, ni.com™, and NI™ are trademarks of National Instruments Corporation. Product and company names mentioned herein are trademarks or trade names of their respective companies. For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or `ni.com/patents`.

September 2003
323452B-01

© 2002–2003 National Instruments Corp. All rights reserved.

NATIONAL INSTRUMENTS™

Complete the following steps to install the LabVIEW DSP Test Integration Toolkit for TI DSP.

1. Verify the following components are installed before continuing with installation:
   - LabVIEW 7.0
   - CCS IDE 2.2 or later
   - TI DSP development platform that you can configure with CCS. The C2000™ DSP platform is compatible only with the toolkit automation VIs.

Refer to the development board documentation for information about installing and configuring the development board.

(Windows 2000/NT/XP) Log in as an administrator or as a user with administrator privileges before you install the LabVIEW DSP Test Integration Toolkit.

2. Insert the LabVIEW DSP Test Integration Toolkit CD into the CD-ROM drive. The LabVIEW DSP Test Integration Toolkit installation program runs automatically.

Note If the setup does not launch automatically, select **Start»Run**, enter *x*`:\setup`, where *x* is the letter of the CD-ROM drive, and click the **OK** button.

3. Follow the instructions that appear on the screen.

The LabVIEW Debugging Workbench for RTDX™ Communication installs with the LabVIEW DSP Test Integration Toolkit. Use the LabVIEW Debugging Workbench for RTDX™ Communication to quickly interact with target code RTDX channels using LabVIEW controls and indicators. Select **Tools»RTDX»LabVIEW Debugging Workbench for RTDX Communication** from CCS or select **Programs»National Instruments»LabVIEW Debugging Workbench for RTDX Communication** from the Windows Start menu to launch the LabVIEW Debugging Workbench.

Automating CCS IDE Functions

Use the LabVIEW DSP Test Integration Toolkit VIs to control the CCS IDE and project files. You can launch CCS IDE, open a `.pjt` file, build the `.pjt` file, and download the resulting `.out` file to a development board from a LabVIEW VI to automate CCS IDE functionality.

The block diagram in Figure 1 shows how you can automate the process of compiling DSP target code and embedding the code on a development board. Wire the `.pjt` file path to the CCS Open Project VI to open the `.pjt` file in CCS IDE. The CCS Build VI builds the `.pjt` file to create the DSP target code `.out` file. The CCS Download Code VI downloads the `.out` file to the development board. The CCS Run VI runs the embedded `.out` file on the development board. You then can use the toolkit RTDX communication and the toolkit memory VIs to access data from the target code.

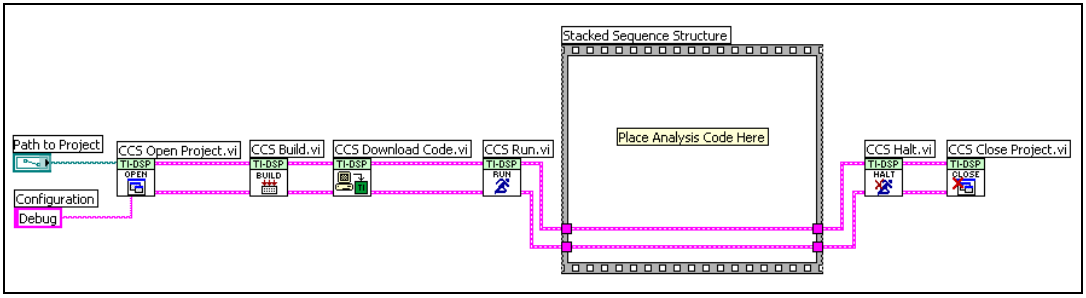

Figure 1. LabVIEW Automation of CCS IDE

The CCS Halt VI and CCS Close Project VI stop the `.out` file running on the development board and close the project in CCS IDE.

Exchanging Data with DSP Target Code

After you create DSP target code in the CCS IDE and embed the `.out` file on a development board, you can create LabVIEW VIs to test the target code. You can transfer simulation data to the DSP target code from a VI and analyze output data from the target code on the development board.

Use the RTDX communication VIs to send data to DSP target code RTDX input channels and to receive data from RTDX output channels. RTDX allows system developers to transfer data between LabVIEW and the target processor without interfering with target code execution. The data inputs can be any test values you want to pass to the target code with each iteration, and the data outputs can be any values you want to retrieve for analysis from the target code after each iteration.

The block diagram in Figure 2 shows how to send data to and receive data from RTDX channels of DSP target code running on a development board. While running, the DSP target code for the example expects sine wave data. The Simulate Signal VI creates a test sine wave. The CCS RTDX Write VI writes the test sine wave to the **WaveIn** RTDX channel of the DSP target code. The DSP target code uses the test sine wave data it receives from the **WaveIn** RTDX channel.

The CCS RTDX Read VI reads waveform data from the **WaveOut** RTDX channel of the DSP target code. This example VI displays the waveform data on a waveform graph.

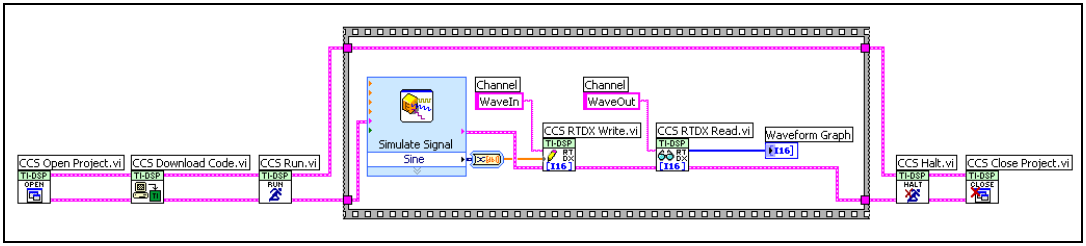

Figure 2. RTDX Communication with DSP Target Code

Writing and Reading Data from DSP Memory

You can access data from the memory of target processors. The memory VIs communicate with the development board slower than the RTDX communication VIs, but you can use them to perform simple code monitoring and control.

The following target code runs on the target processor and executes a loop that increases the value of the `counter` symbol until the `run_flag` symbol equals `0`.

```
volatile Int16 run_flag = 1;
Int16 counter = 0;
void main()
{
    while( run_flag ) {
        // loop executes until run_flag equals zero.
        // counter increases during each iteration.
        counter++;
    }
}
```

In Figure 3, the CCS Symbol to Memory Address VI returns the memory address of the `counter` symbol and passes the memory address to the CCS Memory Read VI. The CCS Memory Read VI reads the value of memory at the memory address and this example VI displays the value of the `counter` symbol on the front panel.

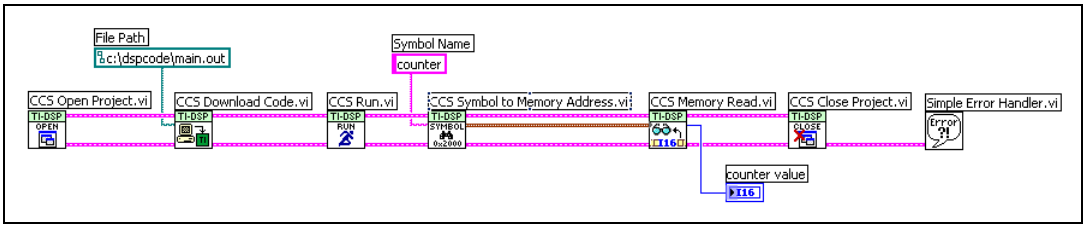

Figure 3. Reading Data Values from Memory

You can control the target code by changing the value of target code symbols. In Figure 4, the CCS Symbol to Memory Address VI passes the current memory address of the `run_flag` symbol to the CCS Memory Write VI. The CCS Memory Write VI changes the value of `run_flag` to `0`, stopping the target code loop.

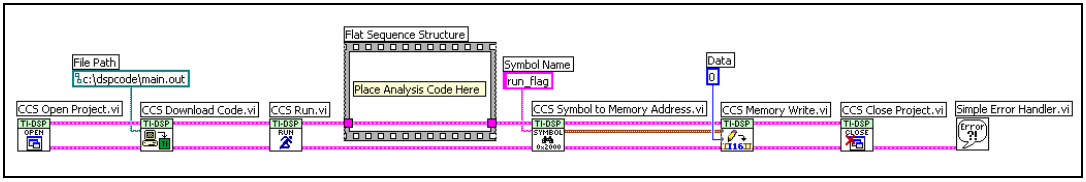

Figure 4. Writing Data Values to Memory

Working with Target Code Data in LabVIEW

You can analyze, manipulate, and present test data the DSP target code returns. For example, on the block diagram in Figure 2, the DSP target code returns waveform test data to analyze in LabVIEW. A waveform graph displays the waveform data. Refer to the *LabVIEW User Manual* for information about the measurement, analysis, and presentation features of LabVIEW.

Using DSP Test Integration Advanced VIs

Denoted by a blue background, the DSP Test Advanced VIs handle low-level CCS IDE and RTDX functionality, such as launching CCS IDE and enabling or disabling RTDX communication. Use the DSP Test Integration Advanced VIs when you want more control over CCS IDE and RTDX functionality.

Use the LabVIEW ActiveX Property and Invoke nodes to control the CCS IDE functionality using the DSP Test Integration Advanced VIs. Refer to the the *LabVIEW User Manual* for information about using Invoke and Property nodes.