LabVIEW™ DSP Test Toolkit for TI DSP

The LabVIEW DSP Test Toolkit for TI DSP uses LabVIEW to automate TI's Code Composer Studio™ (CCS) Integrated Development Environment (IDE) so you can create test systems for DSP target code.

The LabVIEW DSP Test Toolkit includes VIs to automate CCS IDE and VIs that use TI's Real-Time Data Exchange™ (RTDX™) software technology to communicate with TI's DSP Starter Kit (DSK) development boards. Use the toolkit automation VIs to work with CCS IDE project `(.pjt)` files programmatically. Use the toolkit RTDX technology communication VIs to programmatically exchange data with DSP target code.

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

Contents

October 2002
323452A-01

Installation

1. Verify the following components are installed before continuing with installation:
   - LabVIEW 6.1
   - CCS IDE 2.0 or later
   - TI's TMS320C6000™ or C5000™ DSP family platforms (C2000™ DSP platform compatible only with the toolkit automation VIs)

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

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

2. Insert the LabVIEW DSP Test Toolkit CD into the CD-ROM drive. The LabVIEW DSP Test 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 select **OK**.

3. Follow the instructions that appear on the screen.

Automating CCS IDE Functions

Use the LabVIEW DSP Test 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 DSK development board from a LabVIEW test system automating CCS IDE functionality.

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

The block diagram in Figure 1 shows how you can automate the process of compiling DSP target code and embedding the code on a DSK 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 creating the DSP target code `.out` file. The CCS Download Code VI downloads the `.out` file to the DSK development board. The CCS Run VI runs the embedded `.out` file on the DSK development board.

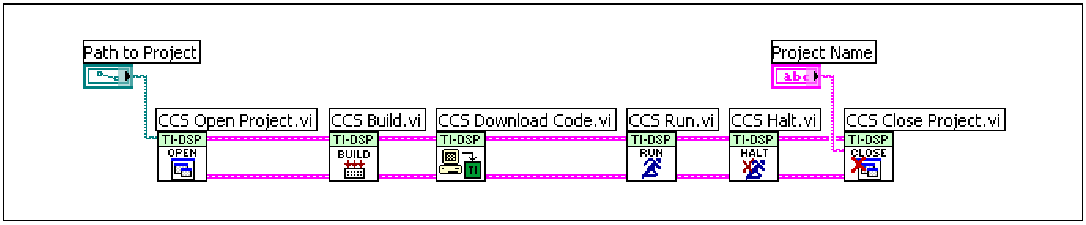

Figure 1. LabVIEW Automation of CCS IDE

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

Exchanging Data with DSP Target Code

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

Use the RTDX technology communication VIs to send data to DSP target code input channels and to receive data from DSP target code output channels. 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.

Refer to the *LabVIEW DSP Test Toolkit Help* by selecting **Help» LabVIEW DSP Test Toolkit Help** for information about the toolkit RTDX technology communication VIs.

The block diagram in Figure 2 shows how to send data to and receive data from DSP target code running on a DSK development board. While running, the DSP target code for the example expects sine wave data. The Basic Function Generator VI creates a test sine wave. The CCS RTDX Write Array I16 VI writes the test sine wave to the **WaveIn** channel of the DSP target code. The DSP target code uses the test sine wave data it receives from the **WaveIn** channel.

The DSP target code returns data without interrupting execution. The CCS RTDX Read Array I16 VI reads waveform data from the **WaveOut** channel of the DSP target code.

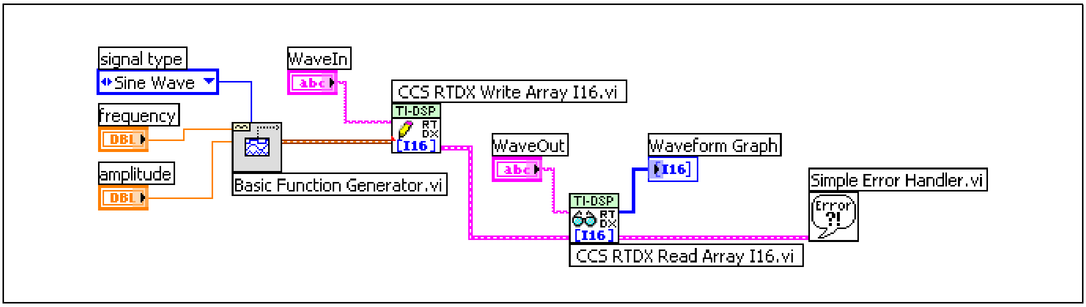

Figure 2. Communication with DSP Target Code

Working with Target Code Data in LabVIEW

You can analyze, manipulate, and present test data returned by the DSP target code. For example, on the block diagram in Figure 2, the DSP target code returns waveform test data to be analyzed 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 Advanced VIs

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

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

The block diagram of the CCS Download Code VI, shown in Figure 3, uses DSP Test Advanced VIs. The CCS Get Build Path VI returns the full path to the compiled `.out` file. The Invoke node receives the `.out` file path and calls a method that downloads the specified `.out` file to the DSK development board. The File/Directory Info function receives the `.out` file path and returns the `.out` file size. Using the `.out` file size, the VI calculates the amount of wait time for the download to complete. During the download, the CCS Status VI updates the user with download status information.

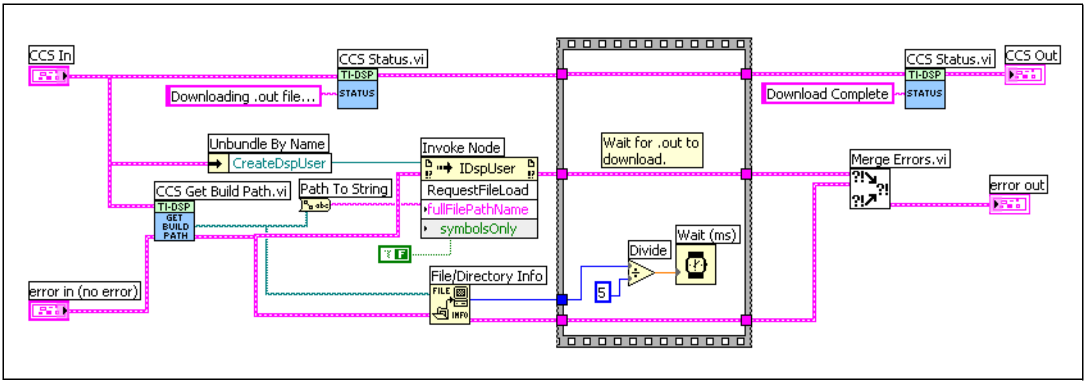

Figure 3. Using DSP Test Advanced VIs

Exploring Examples

The LabVIEW DSP Test Toolkit includes example VIs that demonstrate automating CCS IDE from LabVIEW. The examples also use RTDX technology to communicate between LabVIEW VIs and DSP target code.

Keypad

The Keypad example simulates the functionality of DSP target code that produces cell phone keypad tones. LabVIEW Boolean buttons on the front panel simulate the cell phone keypad. The VI produces signals for each button to test the response of the DSP target code.

Note The Keypad and Keypad Compact examples require TI's TMS320C6711™ or TMS320C6211™ development board with speakers.

Refer to the block diagram when working through the example. Using RTDX technology, the Keypad example sends a value to the **digitIn** channel of the DSP target code. The target code generates the corresponding tone and sends it to the speakers connected to the DSK development board.

This example uses the LabVIEW Event structure to call specific code depending on the keypad button pressed. Refer to the *Event-Driven Programming* section of Chapter 8, *Loop and Case Structures*, of the *LabVIEW User Manual* for information about the Event structure.

Complete the following steps to simulate a cell phone keypad in LabVIEW to test DSP target code.

1. Open and run the Keypad VI example.
2. Click the **Open** button. The CCS Open Project VI calls a CCS IDE function that loads the example project file.
3. Click the **Build** button. The CCS Build VI calls a CCS IDE function that compiles the files in the project and creates the `.out` file.
4. Click the **Download** button. The CCS Download Code VI calls a CCS IDE function that downloads the `.out` file to the DSK development board.
5. Click the **Run** button. The CCS Run VI calls a CCS IDE function that runs the embedded `.out` file on the DSK development board.
6. Click a button on the front panel keypad. A different tone sounds according to the keypad button you click. In the real-world, you can use LabVIEW to analyze the tone and detect problems in the DSP target code.
7. Click the **Halt** button. The CCS Halt VI calls a CCS IDE function that stops the embedded `.out` file running on the DSK development board.
8. Click the **Close** button. The CCS Close Project VI calls a CCS IDE function that closes the `.out` file embedded on the DSK development board.
9. Close CCS IDE and reset power to the DSK development board.

Keypad Compact

Refer to the block diagram when working through the example. The Keypad Compact example performs the same task as the Keypad example. However, the automation VIs are called sequentially on the block diagram instead of individually after the click of a button as in the Keypad example. The automation in the Keypad Compact example is typical to an automated test program.

Complete the following steps to simulate a cell phone keypad in LabVIEW to test DSP target code.

1. Open and run the Keypad Compact VI example. The `.pjt` file opens in CCS IDE, which builds, downloads, and runs the `.out` file on the DSK development board.
2. Click the **Hang up** button to stop the example.
3. Close CCS IDE and reset power to the DSK development board.

Gain

Refer to the block diagram when working through the example. The Gain example uses LabVIEW signal generation to create a waveform and applies a waveform gain factor to the generated waveform. Using RTDX technology, the example sends the waveform to the **cinput** channel of the DSP target code and sends the waveform gain to the **cgain** channel of the DSP target code. The DSP target code combines the original waveform and gain and then returns a scaled waveform to LabVIEW through the **coutput** channel.

Complete the following steps to demonstrate communication between a LabVIEW VI and DSP target code.

1. Open and run the Gain VI example.
2. Select the waveform type from the **Signal Type** enum control.
3. Use the **Gain** slide control on the front panel to add gain to the original sine wave. The front panel waveform graph displays the original waveform and the scaled waveform.
4. Use the **Frequency** slide on the front panel to change the frequency of the signal generated by LabVIEW.
5. Click the **STOP** button to stop the example VI.

Filter

Refer to the block diagram when working through the example. The Filter example uses LabVIEW signal generation to create a waveform signal and LabVIEW filter design to calculate the coefficients for an FIR filter. Using RTDX technology, the example sends the waveform to the **cinput** channel of the DSP target code. When filter parameters change, LabVIEW calculates new filter coefficients and sends them to the **ccoefs** channel of the DSP target code. The DSP target code applies the filter to the waveform and then returns the filtered waveform through the **coutput** channel.

Complete the following steps to demonstrate calculating and sending filter coefficients to DSP target code.

1. Open and run the Filter VI example.

Note Use the `filter_5416dsk.vi` with TI's C5416™ DSP and the `filter_6x11dsk.vi` with TI's C6711™ DSP or C6211™ DSP.

2. Select the waveform type from the **Signal Type** enum control.
3. Use the **Frequency** slide control to change the frequency of the generated waveform.

4. Select a filter from the **filter type** enum control of the **Filter Settings** section. The front panel waveform graph displays the effect of the filter on the waveform.
5. Use the **High Cutoff Frequency** and **Low Cutoff Frequency** slide controls to change the range of the filter. The **High Cutoff Frequency** slide control is used only for bandpass and bandstop filters. Set the **High Cutoff Frequency** higher than the **Low Cutoff Frequency**.
6. Click the **STOP** button to stop the example VI.

RTDX Channel Browser

The RTDX Channel Browser example examines DSP target code embedded on the DSK development board and returns the input or output channel names used by the DSP target code. Refer to the block diagram when working through the example.

Complete the following steps to return channel names used in DSP target code.

1. Open CCS IDE and download DSP target code or any example code to the DSK development board.
2. Run the downloaded code on the DSK development board.
3. Open and run the RTDX Channel Browser VI example.

The RTDX Channel Browser VI displays the input and output channel names of the downloaded code in the **Channel Names** listbox.