Application Report
SLAA326–August 2006

ADS8422 Example Programs

Richard Oed and Lijoy Philipose — *Data Acquisition Products Application*

ABSTRACT

This application report presents a way to interface the ADS8422 to the TMS320VC5510, TMS320C6713, and TMS320C6416 in software and hardware. The hardware solution consists of existing and orderable boards. The software solution demonstrates how to collect conversion results using the provided example programs and is available for download. It also can be used as sample code for users when developing their own software solutions. Project collateral discussed in this application report can be downloaded from the following URL: www.ti.com/lit/zip/SLAA326.

Contents

List of Figures

List of Tables

Trademarks

C5000, C6000, Code Composer Studio are trademarks of Texas Instruments.

1 Introduction

The ADS8422 is a 16-bit, 4-MSPS analog-to-digital converter (ADC) with an internal 4.096-V reference and a pseudo-bipolar, fully differential input. The device is a capacitor-based, successive-approximation register (SAR) converter with an inherent sample and hold. The ADS8422 has 8-bit and 16-bit parallel interface bus options, allowing a variety of processors to interface easily.

This application report presents one hardware and software solution for interfacing this converter with the TMS320VC5510, the TMS320C6713, and the TMS320C6416 digital signal processors (DSP). The software developed uses the direct memory controller present on those DSPs together with the TIMER 1, to collect two blocks of data, each one containing 1024 samples.

2 Hardware

The hardware solutions shown in this report involve either the TMS320VC5510 DSK (DSP starter kit), the TMS320C6713 DSK, or the TMS320C6416 DSK, together with the 5-6K interface board and the ADS8422EVM evaluation module. The hardware used is available and can be ordered from Texas Instruments.

2.1 TMS320VC5510

2.1.1 TMS320VC5510 DSK

The TMS320VC5510 DSK not only provides an introduction to the 'C5500 technology, but also is designed to speed the development of power-efficient applications for audio, video, and data acquisition based on this DSP generation.

For more information, search for part number TMDSDSK5510 on the TI Web site at www.ti.com.

2.1.2 TMS320VC5510 – ADS8422 Hardware Interface

The following is a short description of one of several possibilities to connect the ADS8422 to a TMS320VC5510 DSP. Depending on the user's needs, other ways to connect the converter to the DSP are possible.

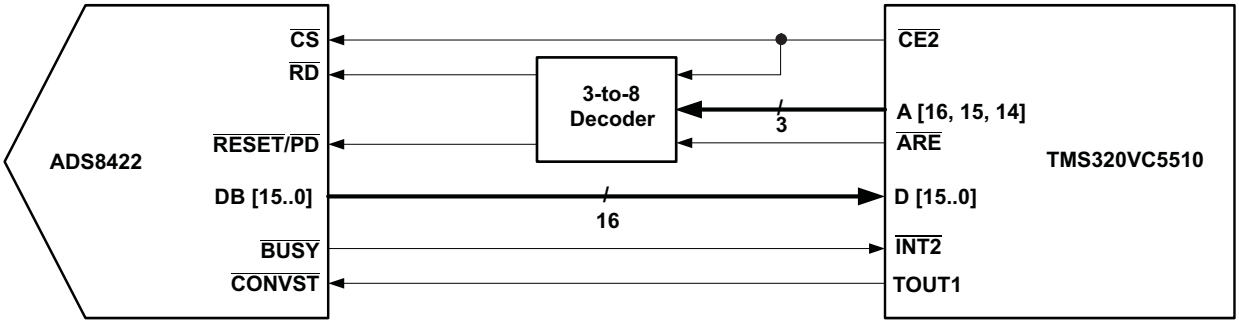

Figure 1. TMS320VC5510 – ADS8422 Hardware Connection

The ADS8422 is mapped into the $\overline{CE2}$ memory space of the TMS320VC5510 DSP. The read and reset signals are generated by using a 3-to-8 decoder on the ADS8422EVM. A read operation from word address 0x416000 generates a pulse on the $\overline{RD}$ pin of the data converter, whereas a read operation from word address 0x41A000 generates a pulse on the $\overline{RESET}/\overline{PD1}$ pin. The $\overline{CE2}$ signal of the DSP acts as $\overline{CS}$ (chip select) for the converter. As the TMS320VC5510 features a 32-bit external memory interface, the BYTE input of the converter can be tied permanently low, disabling the foldback of the data bus.

The $\overline{BUSY}$ signal of the ADS8422 is applied to the $\overline{INT2}$ interrupt input of the DSP, enabling the DMA controller to react on the falling edge of this signal and to collect the conversion result.

The TOUT1 (timer out 1) pin of the TMS320VC5510 is used to source the $\overline{CONVST}$ signal of the converter. Although not all conversion speeds can be generated by the internal timer of the DSP, it is a convenient way to generate this signal.

2.2 TMS320C6713

2.2.1 TMS320C6713 DSK

The TMS320C6713 DSK not only provides an introduction to 'C671x technology, but is powerful enough to use for the fast development of networking, communication, imaging, and other applications like data acquisition.

For more information, search for part number TMDSDSK6713 on the TI Web site at www.ti.com.

2.2.2 TMS320C6713 – ADS8422 Hardware Interface

The following is a short description of one of several possibilities to connect the ADS8422 to a TMS320C6713 DSP. Depending on the user's needs, other ways to connect the converter to the DSP are possible.

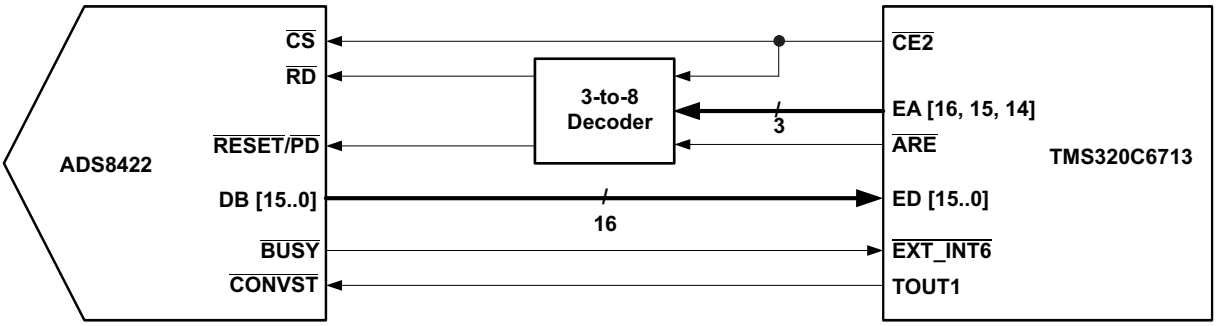

Figure 2. TMS320C6713 – ADS8422 Hardware Connection

The ADS8422 is mapped onto the $\overline{CE2}$ memory space of the TMS320VC6713 DSP. The read and reset signals are generated by using a 3-to-8 decoder on the ADS8422EVM. A read operation from address 0xA000C000 generates a pulse on the $\overline{RD}$ pin of the data converter, whereas a read operation from word address 0xA0014000 generates a pulse on the $\overline{RESET}/\overline{PD1}$ pin. The $\overline{CE2}$ signal of the DSP acts as $\overline{CS}$ (chip select) for the converter. As the TMS320C6713 features a 32-bit external memory interface, the BYTE input of the converter can be tied permanently low, disabling the foldback of the data bus.

The $\overline{BUSY}$ signal of the ADS8422 is applied to the $\overline{EXT\_INT6}$ interrupt input of the DSP, enabling the EDMA controller to react on the falling edge of this signal and to collect the conversion result.

The TOUT1 (timer out 1) pin of the TMS320C6713 is used to source the $\overline{CONVST}$ signal of the converter. Although not all conversion speeds can be generated by the internal timer of the DSP, it is a convenient way to generate this signal.

2.3 TMS320C6416

2.3.1 TMS320C6416 DSK

The TMS320C6416 DSK not only provides an introduction to 'C641x technology, but is powerful enough to use for the fast development of networking, communication, imaging, and other applications like data acquisition.

For more information, search for part number TMDSDSK6416 on the TI Web site at www.ti.com.

2.3.2 TMS320C6416 –ADS8422 Hardware Interface

The following is a short description of one of several possibilities to connect the ADS8422 to a TMS320C6416 DSP. Depending on the user's needs, other ways to connect the converter to the DSP are possible.

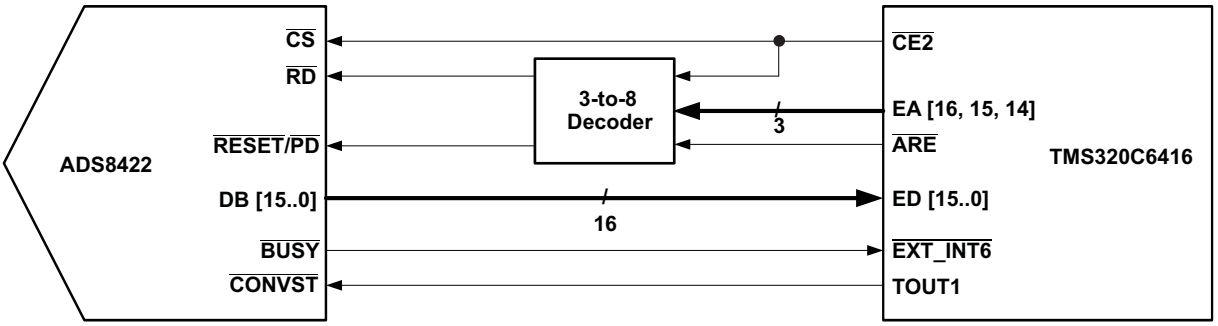

Figure 3. TMS320C6416 – ADS8422 Hardware Connection

The ADS8422 is mapped onto the $\overline{CE2}$ memory space of the TMS320C6416 DSP. The read and reset signals are generated by using a 3-to-8 decoder on the ADS8422EVM. A read operation from address 0xA000C000 generates a pulse on the $\overline{RD}$ pin of the data converter, whereas a read operation from word address 0xA0014000 generates a pulse on the $\overline{RESET}$/PD1 pin. The $\overline{CE2}$ signal of the DSP acts as $\overline{CS}$ (chip select) for the converter. As the TMS320C6416 features a 32-bit external memory interface, the BYTE input of the converter can be tied permanently low, disabling the foldback of the data bus.

The $\overline{BUSY}$ signal of the ADS8422 is applied to the $\overline{EXT\_INT6}$ interrupt input of the DSP, enabling the EDMA controller to react on the falling edge of this signal and to collect the conversion result.

The TOUT1 (timer out 1) pin of the TMS320C64616 is used to source the $\overline{CONVST}$ signal of the converter. Although not all conversion speeds can be generated by the internal timer of the DSP, it is a convenient way to generate this signal.

2.4 ADS8422 Evaluation Module

The ADS8422EVM is an evaluation and demonstration platform for the ADS8422 ADC. The board is a modular, flexible design which allows users to create custom analog signal-conditioning circuits, choose reference sources, and interface modes. It is an easy way to test both the functional and dynamic performance of this 16-bit analog-to-digital converter. The evaluation module includes only those circuits essential to demonstrate the performance of the converter and the interfacing to a parallel bus. These circuits are the analog input, reference, power, digital buffer circuits, and a simple decode logic. The digital inputs and outputs are buffered to isolate the converter from digital noise common to most shared-bus-type systems. The analog-to-digital converter accepts a pseudo-bipolar differential input. A pseudo-bipolar differential signal is a fully differential signal that has a common-mode voltage such that the voltage on each pin is always equal to or above zero volts. See the ADS8422 data sheet (SLAS512) for specific details on recommended input voltages and common-mode range. The positive leg of the input signal can be applied at connector P1 pin 2 (shown in Table 1) or via the center pin of the SMA connector J1. Likewise, the negative input signal can be applied at P1 pin 1 or via the center pin of the SMA connector J2.

The buffered data bus is available via two 0.1-inch IDC header/sockets (P2 and J7). The ADS8422 control inputs also are made available via a standard 0.1-inch IDC header/socket (J2). The decode logic, which generates the read and reset signals, are controlled via connector P3. These standard connectors enable the EVM to be plugged into most prototype boards for rapid evaluation.

The jumper settings needed to operate the EVM together with the desired DSKs are described in the following sections of the example programs.

For additional information on this product and to download the user's guide, search for the part number ADS8422EVM on the TI Web site www.ti.com.

2.5 5-6K Interface Evaluation Module

Many data acquisition evaluation modules (EVM) from Texas Instruments have a common set of connectors and signals at those connectors. The 5–6K interface board allows designers to easily connect those EVMs to the C5000™ and C6000™ family of digital signal processor starter kits.

The 5–6K interface board consists of two serial connectors, two signal conditioning areas, and a parallel interface. The ADS8402EVM plugs into connectors J10 (analog), J17 (data bus), J18 (control bus), and JP5 (power). More information is available in the 5–6K Interface Board User's Guide (SLAU104) , or search the TI Web site for keyword *5–6K interface*.

Table 1 lists the jumper settings for the 5–6K interface board for the different DSP platforms.

Table 1. Jumper Settings for the 5-6K Interface Board

Designator	TMS320VC5510	TMS320C6713	TMS320C6416
W1	OPEN	OPEN	OPEN
W2	2-3	2-3	2-3
W3	2-3	2-3	2-3
W4	2-3	2-3	2-3
W5	2-3	2-3	2-3
W6	1-2	1-2	1-2
W7	2-3	2-3	2-3
J13	INTb	INTb	INTb
J14	TOUTb	TOUTb	TOUTb
All others	don't care	don't care	don't care

3 Software Interface

The example interface software demonstrated in this section is based on the code created by the data converter support tool in Code Composer Studio™. This tool allows the automatic generation of interface software for various data converters from Texas Instruments. The following code was created with the version 4.00 of the tool and tested on Code Composer Studio 3.1. All screen shots and menu descriptions are in reference to those versions.

The example projects can be downloaded from TI's Web site under the *Download associated code files* on the same page this application report is located.

For more information about the data converter support tool, visit www.ti.com/dcplug-in.

3.1 Theory of Operation

The following example programs all follow the same steps to collect two blocks of 1024 samples each from the ADS8422. The flowchart in Figure 4 illustrates these steps.

- Initialization occurs of the global variables, the timer 1, the EDMA channel (DMA channel for the 'C5510), and the external memory interface (EMIF) by calling the function *dc_configure()* with the settings of the ADS8422. These settings are stored in the *ADS8422_1* data structure (defined in *t8422_fn.h*) which contains the information about the physical settings used by the interface software. If the initialization fails for any reason, an error message appears in the stdout window of the Code Composer Studio™.
- The output signal TOUT1 generated by the timer of the DSP, which was configured during the call to *dc_configure()*, is used as input to the $\overline{\text{CONVST}}$ pin of the ADS8422. The interrupt generated by the timer is used on the TMS320C6713 and the TMS320C6416 as a trigger for the EDMA channel to collect the samples from the converter; the $\overline{\text{BUSY}}$ signal is ignored by this setup. This ensures that the required quiet times around the falling edge of the $\overline{\text{CONVST}}$ are met. On the TMS320C5510, this approach is not feasible and the falling edge of $\overline{\text{BUSY}}$ is used to generate the trigger to the DMA controller.
- Once the configuration is complete, the interrupts are enabled globally.
- After checking for an ongoing transfer (for the first iteration of the loop, there is none), a first block of data is requested with a call to the *dc_readblock()* function (residing in the file *t8422_ob.c*), with the settings of the ADS8422, the address of the memory block where the data should be written to, the number of samples to be collected, and a pointer to a callback function. This callback function is invoked once the transfer is complete, signaling the availability of new data. This is necessary, as the

dc_readblock() function returns immediately once the transfer was submitted to the (E)DMA controller.

- Once the *dc_readblock()* function returns to the main() routine, the program sets the global variable *iAdsBusy* to one, signaling that one data transfer is ongoing. This variable is decremented inside the callback function once the transfer is complete.
- After that, the software checks if the maximum number of transfers has been reached (comparing *iAdsBusy* to *ADS8422_MAX_BLOCKS*). If yes, it waits until one transfer is complete and continues.
- Now, a second block of data is requested with another call to the *dc_readblock()* function. If the previous transfer is not yet complete, this transfer is queued inside the (E)DMA controller and is executed once the previous transfer has finished. Note that the software is limited to one ongoing and one queued transfer.
- After the return of the *dc_readblock()* function, the variable *iAdsBusy* is incremented again.
- Once a transfer has finished, the interrupt service routine of the (E)DMA controller calls the callback function, which decrements the global variable *iAdsBusy* by one, signaling the end of a transfer to the application.

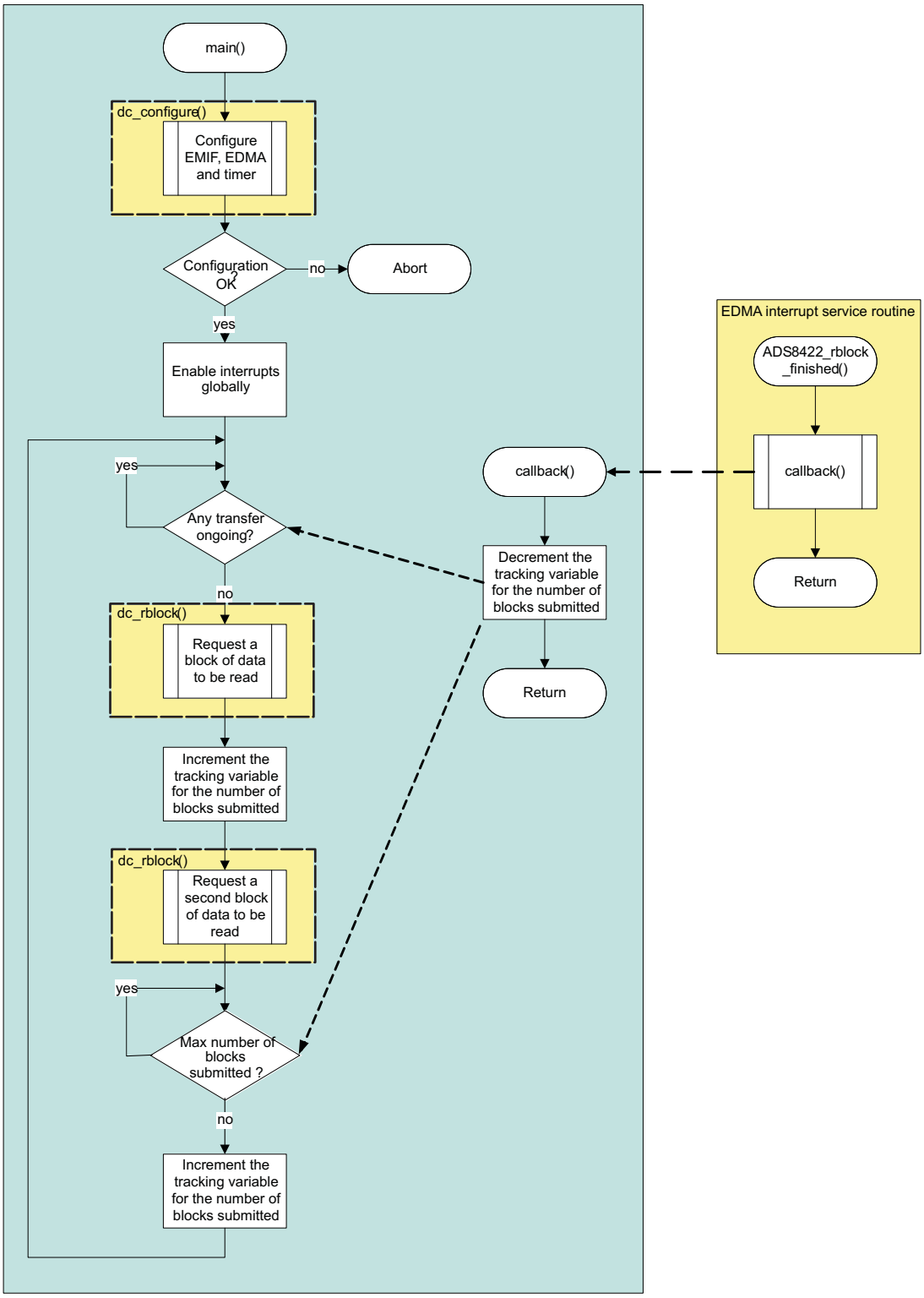

Figure 4. Flowchart of the Application

3.2 Preparing the Hardware

The following hardware is required to run the example code:

- Either one of TMS320VC5510 DSK, TMS320C6713 DSK, or TMS320C6416 DSK

- 5-6K interface board
- ADS8422EVM

In order to get the expected results, the following steps need to be performed:

- Ensure that the jumpers on the 5-6K interface boards are set according to the Table 1.
- Ensure that all jumpers on the ADS8422 EVM are set as outlined in the Table 2.

Table 2. ADS8422 EVM Jumper Settings

Designator	TMS320VC5510	TMS320C6713	TMS320C6416
W1	1-2	1-2	1-2
W2	2-3	2-3	2-3
W3	1-2	1-2	1-2
W4	2-3	2-3	2-3
W5	OPEN	OPEN	OPEN
W6	1-2	1-2	1-2
W7	2-3	2-3	2-3
W8	1-2	1-2	1-2
W9	1-2	1-2	1-2
W10	1-2	1-2	1-2
W11	1-2	1-2	1-2
J4	1-2	1-2	1-2

- Plug the 5-6K interface board onto the DSK of your choice.
- Plug the ADS8422EVM onto the 5-6K interface board.
- Apply ±12 V to J1 on the 5-6K interface board.
- Apply +5 V to J2 on the 5-6K interface board.
- Apply power the DSK.
- Apply an analog signal on J1. For example, apply a 3.5-kHz sine wave with 3 Vpp and 0-V offset.

3.3 Working With the Software

Once the hardware setup is complete, the next step is to install the software and to start Code Composer Studio™:

- Once you have downloaded the associated code files, unzip the files to the folder C:\CCStudio_v3.1\MyProjects. This step creates the following three subdirectories:
  - **ADS8422-C5510-CCS3v1:** Contains the example project for the TMS320VC5510 DSK.
  - **ADS8422-C6713-CCS3v1:** Contains the example project for the TMS320C6713 DSK.
  - **ADS8422-C6416-CCS3v1:** Contains the example project for the TMS320C6416 DSK.
- Now start Code Composer Studio™ (CCS).
- In Code Composer Studio™, open the project for your DSP hardware platform (see Figure 5 and Figure 6) by clicking on *Project* → *Open* on the Code Composer Studio™ menu bar and loading the respective project for your hardware/starter kit from the path applying to your hardware:
  - For the TMS320VC5510 DSK: *Open ADS8422-C5510-CCS3v1.pjt*
  - For the TMS320C6713 DSK: *Open ADS8422-C6713-CCS3v1.pjt*
  - For the TMS320C6416 DSK: *Open ADS8422-C6416-CCS3v1.pjt*

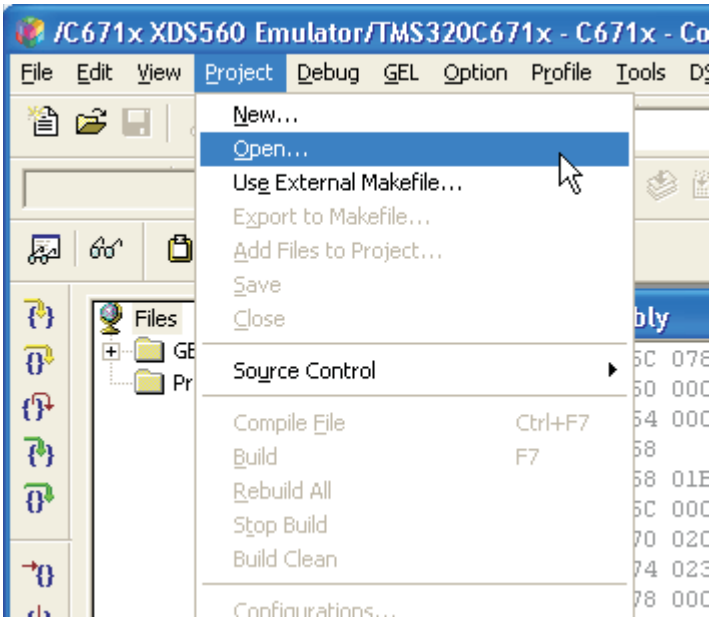

Figure 5. CCS Project Menu

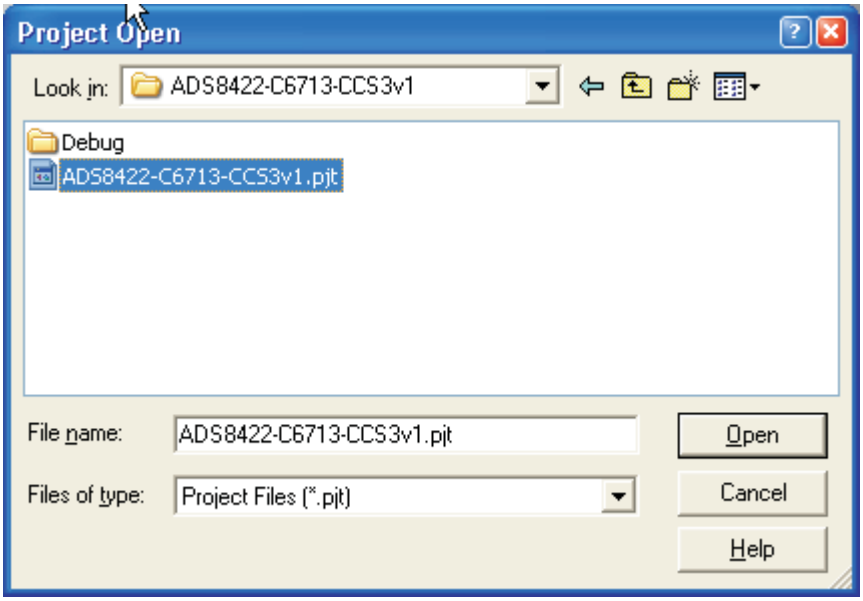

Figure 6. CCS Project Open Dialog

- Once the project is open, expand the project tree in the project view window of CCS by clicking on the **+** (plus sign) in front of it, and expand the Source files as well. The project window should look like the one in Figure 7.

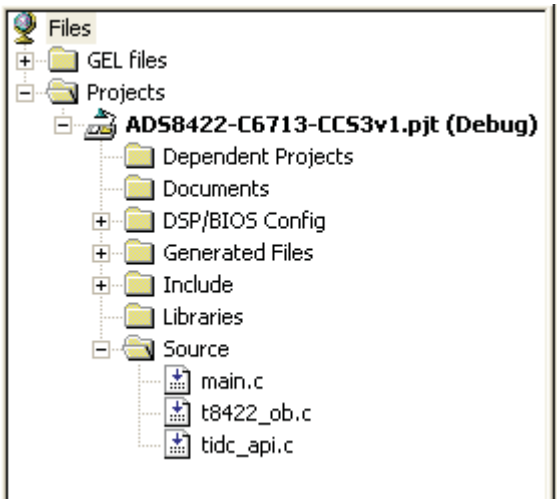

Figure 7. CCS Project View

- The next step is to load the executable program provided into the DSP by clicking on *File → Load Program* and selecting the file *ADS8422-Cxxxx-CCS3v1.out* from the *Debug* subfolder, where xxxx in the filename stands for the platform you are using ('5510, '6713, or '6416).
- Now open the main.c source file by double-clicking on the file name in the project view and scroll down to the line, where the first read block command for the ADS8422 is issued (*dc_readblock() function*). This line reads:

```
 while(1) /* loop forever */ { while(iAdsBusy >= 1); /* transfer in progress? */ /* issue a read
block command to the converter */ dc_readblock(&Ads8422_1, /* data converter object */ r_buffer1,
/* address of the target buffer*/ BUFFER_SIZE, /* size of the target buffer */ callback); /*
callback function */ iAdsBusy++;
```

- Set the cursor in front of the *dc_readblock()* command and right-click in this line. A new pop-up menu opens; select *Insert Graph* (see Figure 8).

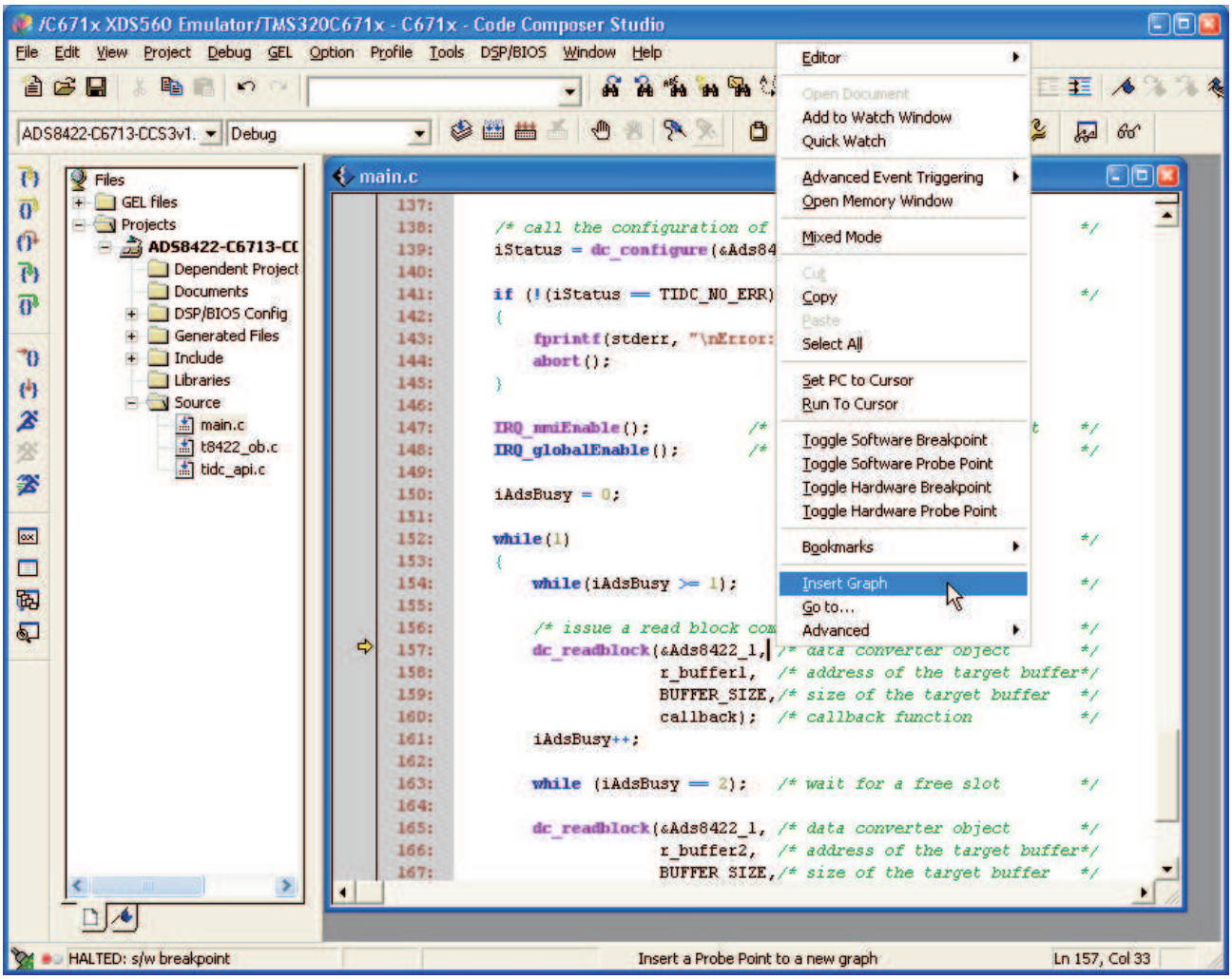

Figure 8. CCS Context Menu

- With this done, a new dialog window with the properties of the graph appears (see Figure 9). Here a few values must be entered, whereas the most settings can be left in the default position:
  - **Start Address:** This is the starting location of the acquisition buffer containing the data to be graphed. When the graph is updated, the acquisition buffer, starting at this location, is fetched from the memory of the DSK. The acquisition buffer then updates the display buffer, which is graphed. In the case of the sample program, set it to **r_buffer1**.
  - **Acquisition Buffer Size:** This is the size of the acquisition buffer you are using on your target board. In the case of the sample program, the size of the *r_buffer1* is defined with the symbol *BUFFER_SIZE* in the *main.c* file and should be set to **1024** in the dialog window.
  - **Display Data Size:** This is the size of the display buffer that you use. The content of the display buffer is graphed on your screen. The display buffer resides on the host; so, a history of your signal can be displayed even though it might no longer exist on the target board. In the case of the sample program, set it to **1024** as well.
  - **DSP Data Type:** Select *16-bit signed integer* form the drop-down list, as the ADS8422 is a 16-bit converter.
  - **Sampling Rate (Hz):** This field contains the sampling frequency of the converter. The sampling rate is used to calculate the time and frequency values displayed on the graph. For a time domain graph, as used in this example, this field calculates the values for the time axis. The axis is labeled form 0 to (Display Data Size * 1/Sampling Rate). For the sample program, set it to the following values depending on the DSP platform you are using:

- **TMS320VC5510:** 4000000 Hz
- **TMS320C6713:** 3515625 Hz
- **TMS320C6416:** 3750000 Hz

More information on the different settings can be found in the help file of Code Composer Studio™.

Once all settings are entered, click on the *OK* button to close the dialog. The graph window appears, currently with no useful values, as the program was not started yet.

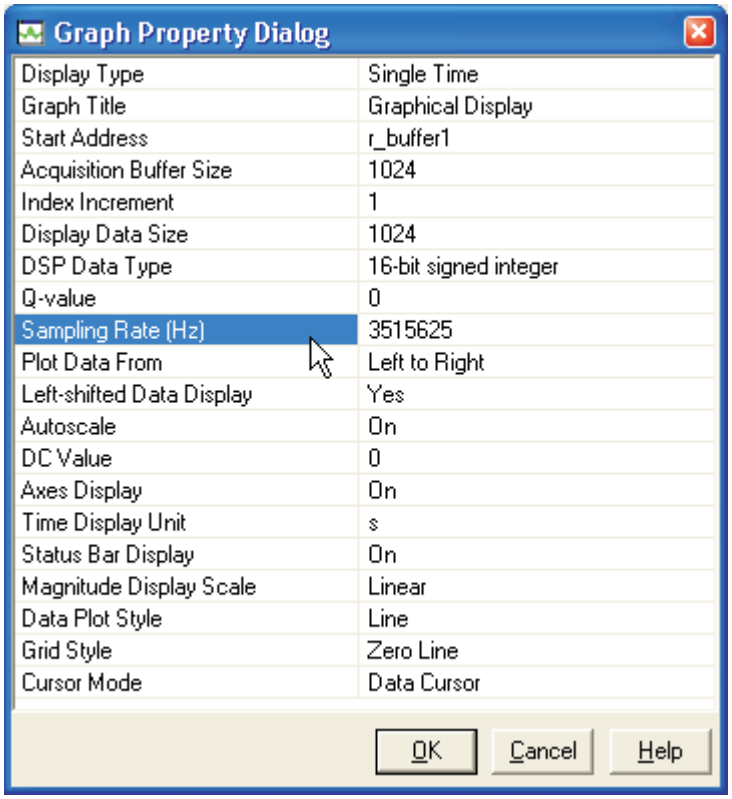

Figure 9. CCS Graph Property Dialog

- Start the program by pressing the <*F5*> key on your keyboard. After a short time, the graph window is updated and shows the waveform of the analog signal (see Figure 10).
- If the second buffer should be displayed as well, the preceding steps should be repeated, with the variable *r_buffer2* instead of *r_buffer1* for the *Start Address*. If you wish, you can set the *Graph Title* property to *Buffer 2* (or any other meaningful name) to make it easier to distinguish between the two graphs.

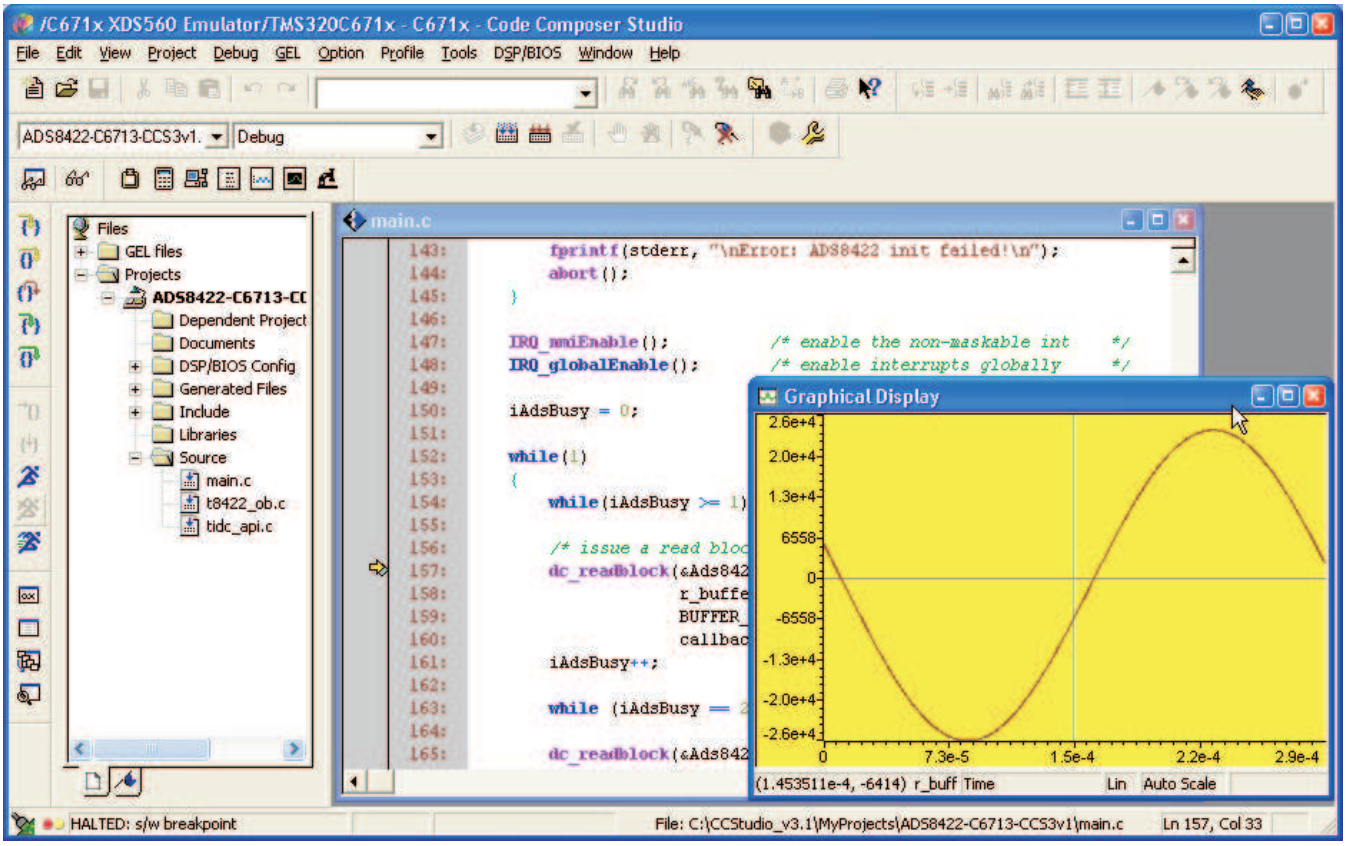

Figure 10. CCS Window With Graph

4 Conclusion

This application report presents one solution to interface the ADS8422 to the TMS320VC5510, TMS320C6713, or TMS320C6416 DSPs in software and hardware. All the hardware necessary for this application report can be ordered directly from Texas Instruments, making this solution a relatively inexpensive hardware design. Also, the example programs are available from Texas Instruments as a free download.

As the timer of the DSP was used to generate the $\overline{\text{CONVST}}$ signal, it was not possible to operate the converter at full speed on the TMS320C6713 and TMS320C6416. However, in a real-world design, the $\overline{\text{CONVST}}$ signal would be generated by an external clock source, so this would be no limitation.

If the software needs to be adapted to a different hardware setup, the Data Converter Support Tool in Code Composer Studio™ is a great help. The newest version of this tool can be downloaded from www.ti.com/dcplug-in and once installed, can be accessed from inside Code Composer Studio™ by selecting Tools → Data Converter Support. A separate application report, *Getting Started with the Data Converter Plug-In* (SLAA210), describes how to use this tool and gives more information on the functions used in the example projects used.

5 References

1. *ADS8422, 16-Bit, 4-MSPS, Pseudo-Bipolar, Fully Differential Input, Micropower Sampling Analog-to-Digital Converter With Parallel Interface, Reference data sheet* (SLAS512)
2. *5-6K Interface Board User's Guide* (SLAU104)
3. *Getting Started With the Data Converter Plug-In* (SLAA210)

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265