Interfacing the ADS8345 to TMS320C5416 DSP

Lijoy Philipose *DACQ Applications Groups*

ABSTRACT

This application report presents a hardware and software solution to interfacing the ADS8345 16-bit, 8-channel, 100 kHz analog-to-digital converter to the C5000™ fixed point 16-bit family of digital signal processors (DSP). While this application report primarily focuses on the 8-channel ADS8345 part, the same solution can be utilized for the 4-channel ADS7841/ADS8341/ADS8343 and 8-channel ADS8344/ADS7844 devices.

The hardware interface employed is the ADS8345EVM, C5000™ & C6000™ DAP interface card and the TMS320C5416™ DSP starter kit (DSK), all provided by Texas Instruments.

The software interface was written using the chip support library (CSL) and DSP/BIOS functions. These features of the Code Composer IDE were used to program and communicate with the multichannel buffered serial port (McBSP) and the DMA controllers.

The software developed is available for download from TI's website. Search on this application report literature number (SLAA160). Project collateral discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/SLAA160.

Contents

Figures

1 Introduction

The ADS8345 is an 8-channel, bipolar input, 16-bit sampling analog-to-digital (A/D) converter with a synchronous serial interface. It can be interfaced easily to the TMS320C5416™ digital signal processor via the multichannel buffered serial port (McBSP) peripheral. Presented in this report is a hardware and software solution for interfacing the ADS8345 to C5416™ DSP. The hardware solution describes physical connections between the A/D and DSP. The software solution comes in the form of C language code written to capture 2048 samples from the A/D. The software solution utilizes the McBSP and DMA peripherals. The McBSP1 is used as the serial port. The DMA controller transfers configuration and data between DSP memory and the A/D converter. One DMA channel writes the command word out to the A/D, while another reads digitized words in from the A/D. Using the DMA controller to transfer data in and out of the A/D allows the CPU to perform other tasks until the DMA triggers an interrupt indicating A/D data has been collected and is ready for processing.

We begin by describing the hardware interface. The final sections of this application note focus on the interface and associated software. The software written for this note is available online for download.

2 Hardware Interface

Texas Instruments provides all the hardware necessary to quickly prototype this system. Three pieces of hardware are used to complete the interface—the ADS8345EVM, TMS320C5416™ DSK and the C5000™ & C6000™ DAP Interface Card.

2.1 ADS8345EVM

The ADS8345EVM provides a platform for quickly prototyping the 16-bit 8-channel 100 kHz ADS8345 analog-to-digital converter in any system by providing standard 0.1-inch headers to all the digital I/O lines and analog inputs.

The P1 connector, on the left side, is the connector that applies the analog input signals to the converter. The first channel can be buffered via IC U1; all other signals are applied directly to the converter through a low pass filter.

The P2 connector, on the right side, is the connector that applies the digital signals to the ADS8345. These signals are directly applied to the converter; therefore care must be taken externally to prevent significant overshoot and undershoot of these signals. Significant overshoot and undershoot of the digital signals results in increased offset error.

U3 is the reference chip that provides a steady 2.5 V, via buffer amplifier U5.

Board power is applied through J3. This board accepts ±12 V and +5 V, and uses the ±12 V to power the reference chip, and set the operational amplifier rails. The 5-V supply is applied to the ADS8345 and the digital output buffers (U6 & U7). For more information, see the ADS8345EVM user's manual (SBAU082).

2.2 TMS320C5416™ DSK

The C5416™ DSK is a low-cost development platform designed to cut development time. The kit, which provides new performance-enhancing features such as USB communications and true plug-and-play functionality, gives both experienced and novice designers an easy way to get started immediately with innovative product designs. Check TI's website for more information.

2.3 C5000™ & C6000™ DAP Interface Card

This interface card is a simple connector board. It sits between the analog-to-digital converter evaluation board and the DSP starter kit. It brings out a multichannel buffered serial port, an interrupt pin, and a general-purpose pin from the 80-pin SAMTEC connectors of the DSK to a 20-pin IDC serial connector. Check the TI's website for more information.

2.4 Hardware Connections

The hardware connections between the C5416™ and ADS8345 are as shown in Figure 1. CLK, BUSY, DIN and DOUT are wired to CLKX1, FSR1, DXR1 and DRR1 respectively. The ADS8345 is the only device on this serial bus; therefore chip select is hard-wired to ground. If multiple devices are attached to this bus, chip select cannot be grounded. In this case, CS must tie to a GPIO pin or other decode logic to ensure that no two devices access the bus at one time.

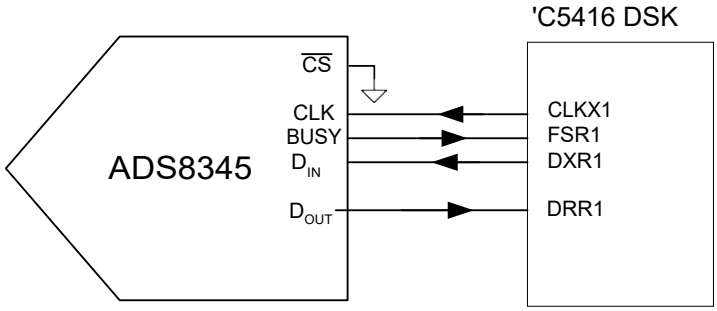

Figure 1. Hardware Interface

3 Software Interface

The software interface is written entirely in C under Code Composer Studio™ (CCS) IDE using CSL and DSP/BIOS. This report assumes readers have a basic understanding of DSP/BIOS and CSL as found in the help system of CCS. If not, TI recommends the user go through the tutorials in Code Composer Studio™ before proceeding further.

First let us state the design objective: Collect 256 samples sequentially from all 8 channels of ADS8345. The DSP can be used for initialization and data processing, and not for each data point collection.

Since the DSP cannot be used to read and write to the A/D, we use the DMA channels to accomplish this task. The C5416™ has six DMA channels. We use two of them in auto buffer mode. DMA channel 4 (DMAC4) writes command bytes (from array *CfgBtye*) to the transmit register of McBSP1. DMA channel 5 (DMAC5) reads from the receive register of McBSP1 and stores digitized code into memory (to array *buffer*). Once all 256 samples from all 8 channels have been read in and stored, DMAC5 interrupts the DSP indicating the data is ready for processing. When the interrupt is received, the DSP disables further transfers and sorts the interleaved channel data from *buffer* and stores them in channel arrays. Data from the different channels are sorted and stored in arrays called Chan0, Chan1, Chan2, Chan3, Chan4, etc.

As mentioned above, the software and hardware interface utilizes McBSP1 and DMA peripherals, as well as the DSP/BIOS and CSL. All the registers for both DMA and McBSP are set using the configuration tool via selection tab and pull down options. The following sections describe how this is done.

The first step is to open the configuration tool (double click on the file that ends with .CDB). The following describes DSP memory allocation (requirement for DMA auto buffer mode), setting McBSP and DMA register using CSL, and finally how to set up hardware interrupt and software interrupts using DSP/BIOS

3.1 DSP Memory Allocation

In order to use the DMA in auto buffer mode, the buffer base address must be based on powers of two. To satisfy this requirement the array (*CfgBtye*) of length 8 base address must be aligned with 8 word boundaries (ex. 0x0000, 0x0008, 0x0010, 0x0018). The array buffer, which is 2048, must be aligned with 2048 word boundaries (0x800, 0x1000, 0x1800).

For this report, *CfgBtye* is arbitrarily placed at 0x5FF0 and *buffer* at 0x06000. The configuration seed file for the C5416DSK allocates IDATA from 0x000080 to 0x007080. Before creating memory sections for *CfgByte* and *buffer*, IDATA section must be redefined.

1. Expand System module in the configuration tool. Right click on IDATA and select properties as shown in Figure 2. Change *len* property to 0x5F70 as shown in Figure 3.

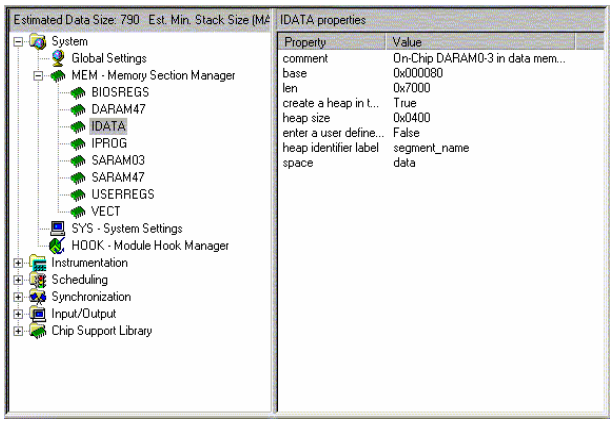

Figure 2. IDATA Section Selection

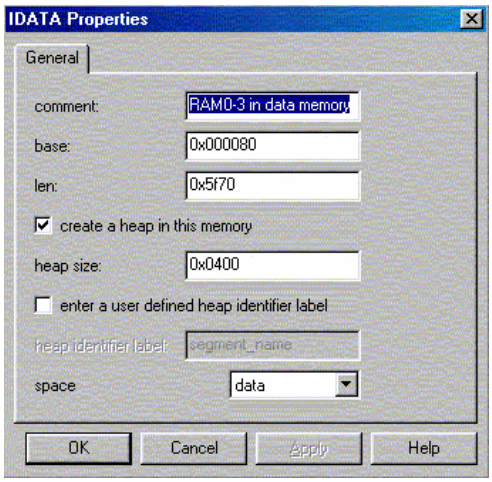

Figure 3. IDATA Section Remapped

2. Right click on MEM-Memory Section Manager and select Insert MEM. Repeat this step once more. Rename the first object to DMAMEM and second to DMAMEM1 as shown in Figure 4.

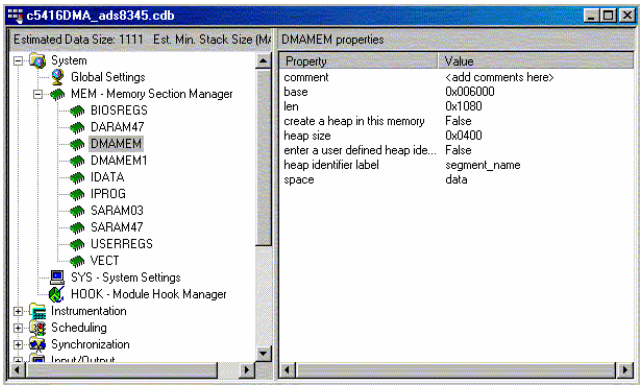

Figure 4. DMAMEM and DMAMEM1 Objects

3. Right click on DMAMEM and set properties as shown in Figure 5. Notice the *len* property is 0x1080 (4224d). Since the DMAMEM section is 4224 words long, the buffer size can be up to 4224 words long.

4. Right click on DMAMEM1 and set properties as shown in Figure 6. Notice that here also the *len* property is set to 0x10 (16d). Since the DMAMEM1 section is 16 words long, the buffer size can be up to 16 words long.

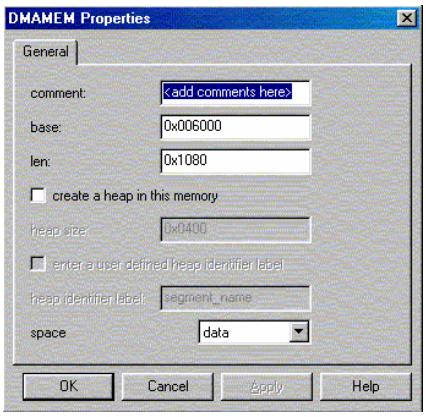

Figure 5 DMAMEM Section Defined

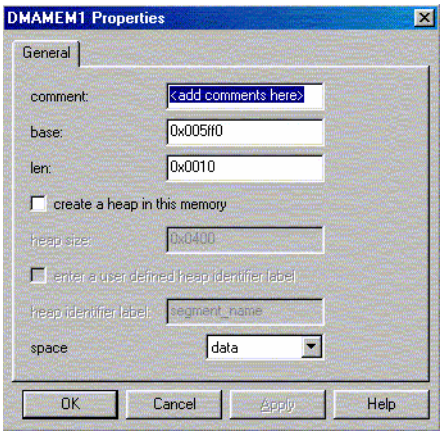

Figure 6 DMAMEM1 Section Defined

The memory sections are now mapped into appropriate boundaries. The pragma DATA_SECTION(,) in the main file places array *buffer* in dmaMem and *CfgByte* in dmaMem1 sections.

```
#pragma DATA_SECTION(buffer, "dmaMem")
Uint16 buffer[NSAMPLES];                    /* A/D Data array */
#pragma DATA_SECTION(CfgByte, "dmaMem1")
Uint16 CfgByte[NConfigWord];                /* A/D commands array */
```

3.2 McBSP Settings

The McBSP is easily programmed using the configuration tool. Implement the following steps to create a McBSP configuration object.

1. Right click on McBSP Configuration Manager and select *insert mcbspCfg*. You may choose to rename the object to mcbspcfg1, since this object is used to configure McBSP1.

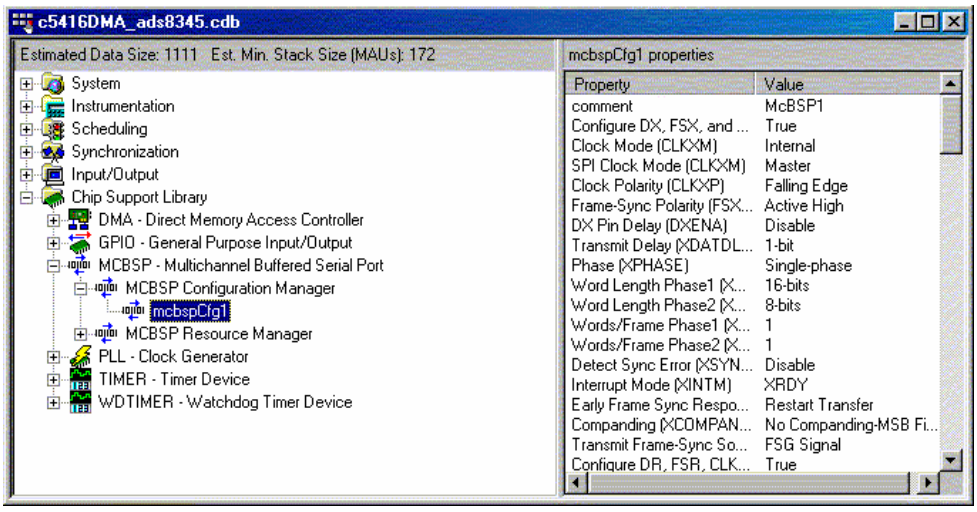

Figure 7. McBSP Configuration Object

2. Right click on mcbspCfg1 and select properties. The properties screen shows three rows of tabs allowing the reader to select the McBSP settings for transmitter, receiver, sample-rate, pin control, etc.

3. In the *General* tab, select options as shown in Figure 8. Configure McBSP as a DSP serial port.

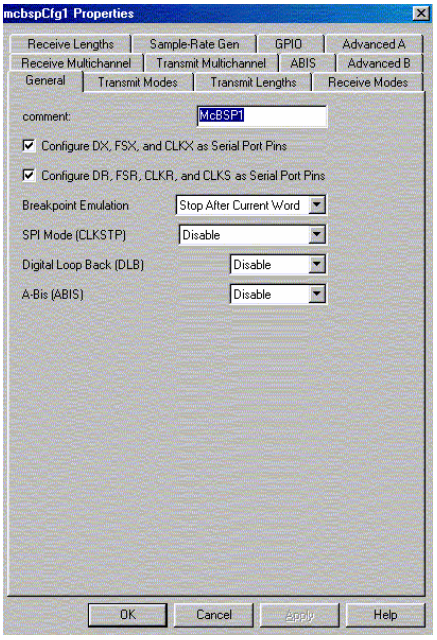

Figure 8. McBSP General Tab

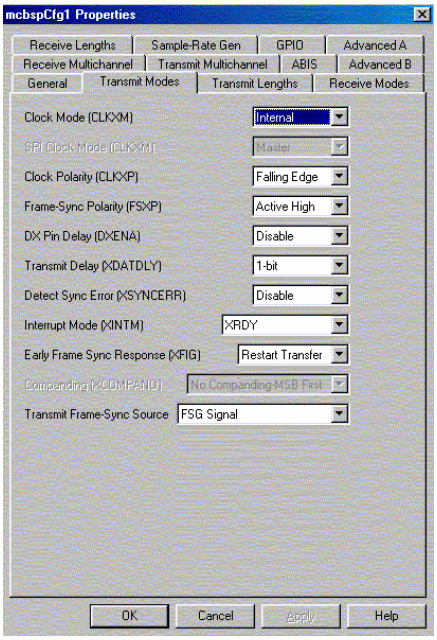

Figure 9. McBSP Transmit Mode Tab

4. In the *Transmit Modes* tab, select options as shown in Figure 9. Set CLKX and FSX to be generated by McBSP. Set CLKX and FSX polarity to falling edge and active high, respectively. The transmit delay bit field is set to 1-bit, causing the START bit to appear on DX following the falling edge of unconnected FSX1 signal.

5. In the *Transmit Length* tab, select options as shown in Figure 10. Set for single-phase, 16-bit transfers.

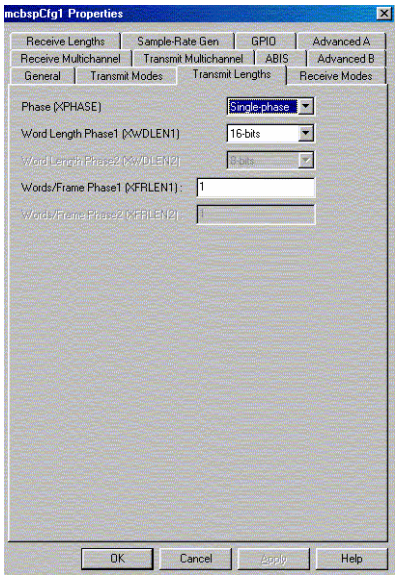

Figure 10 McBSP Transmit Length Tab

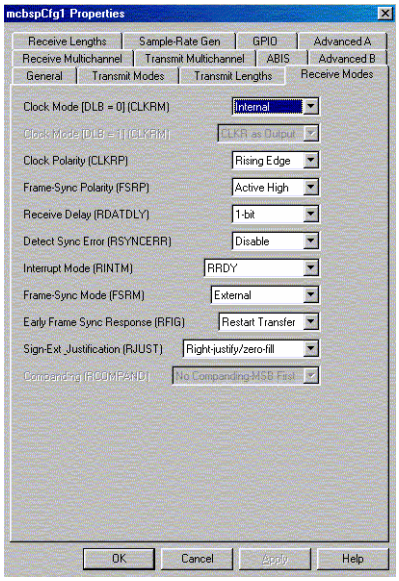

Figure 11. McBSP Receive Modes Tab

6. In the *Receive Modes* tab, select options as shown in Figure 11. McBSP generates CLKR internally. It is in sync with CLKX, so CLKR does not need to be connected. Set CLKR and FSR polarity to rising edge and active high respectively. FSR is generated by an external device; in this case it is the BUSY pulse of the ADS8345. The receive delay is set to one because the A/D shifts data out on the next clock after BUSY returns low.

7. In the *Receive Length* table, select options as shown in Figure 12. Set for single-phase 16-bit transfers. In the *Sample-Rate Gen* tab, set the options shown in Figure 13. Clock source is CPU clock. Frame width is 1. In this application the ADS8345 should run at max sample rate. To achieve 100 kHz sample frequency on C5416™ DSK, clock divide must be 64 and frame period 24.

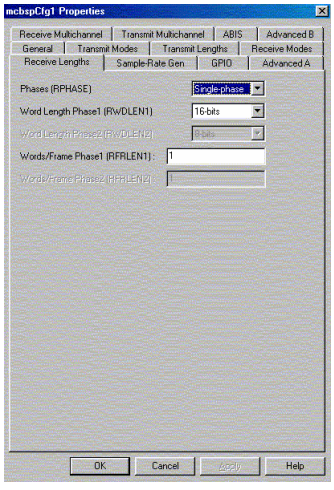

Figure 12. McBSP Receive Length Tab

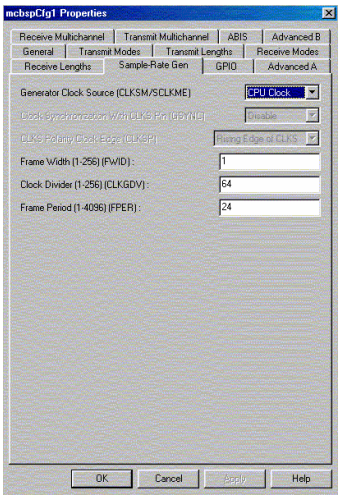

Figure 13. McBSP Sample Rate Generator Tab

8. Now that the McBSP configuration object is properly set, it is time to link the object with the McBSP1 resource. Expand McBSP resource manager. Right click on MCBSP1 and select properties. Set options as shown in Figure 14. Check *open handle* and *pre-initialize* to mcbspcfg1. The pre-initialize ensures that the McBSP registers are set before arriving in the main function.

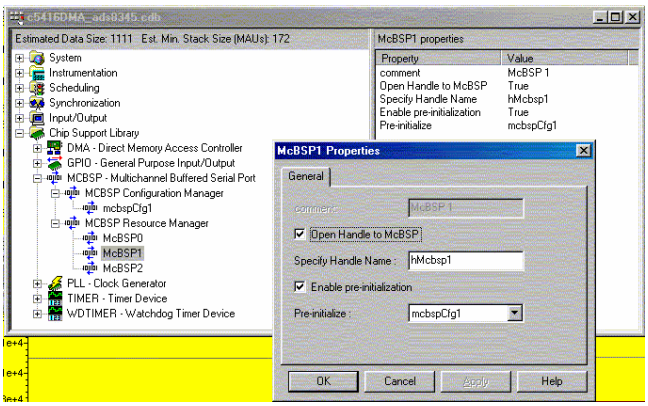

Figure 14. McBSP Configuration Object Resource Link

3.3 DMA Setup

The DMA is easily programmed using the configuration tool. Implement the following steps to create a DMA configuration object. As mentioned above, two DMA channels are used—one channel writes to the McBSP and another channel reads from the McBSP.

1. Right click on DMA configuration manager and select *insert dmaCfg* twice. This creates two objects called dmaCfg0 and dmaCfg1. DMA channel 4 and 5 are utilized for this application, so rename the objects to dmacfg4 and dmacfg5, as shown in Figure 10.
2. Right click on dmaCfg4 and select properties. In the *General* tab you may choose to add the comment s*end channel*, as shown in Figure 16.

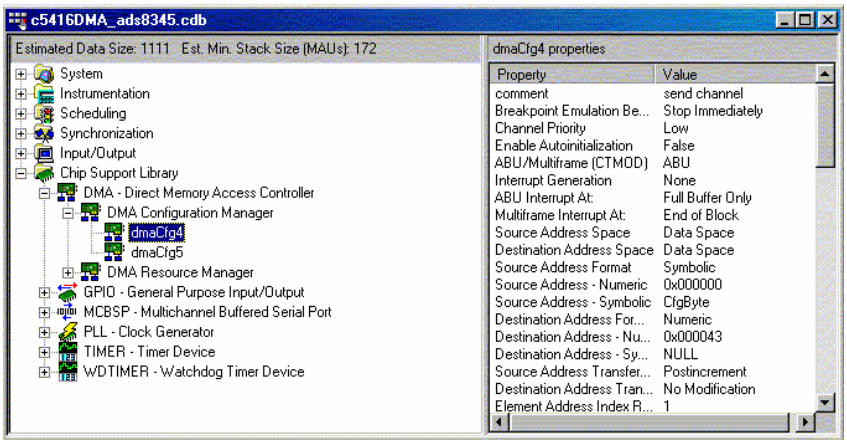

Figure 15 DMA Configuration Objects

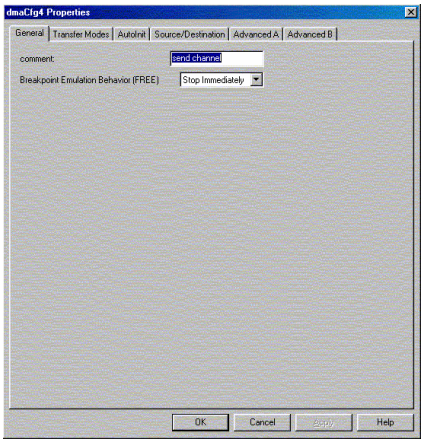

Figure 16. DMA dmaCfg4 General Tab

3. In the *Transfer Modes* tab, set options as shown in Figure 17. Select *Auto Buffer Mode* and synchronize to McBSP transmit event.

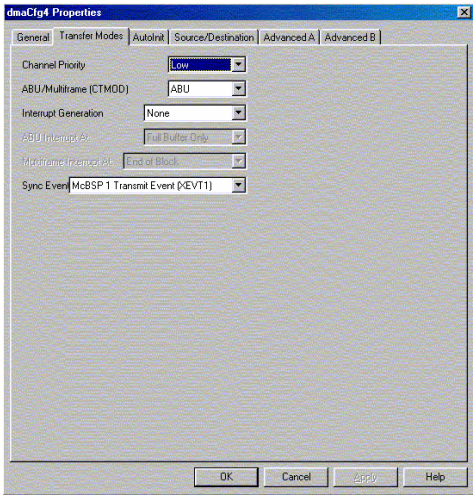

Figure 17. DMA dmaCfg4 Transfer Mode Tab

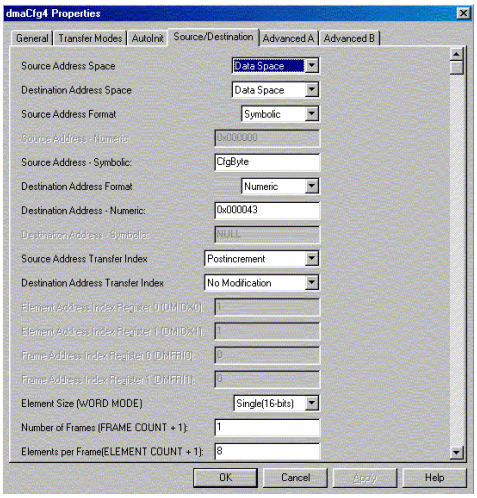

Figure 18. DMA dmaCfg4 Source/Destination Tab

4. In the *Source/Destination* tab, set options as shown in Figure 18. Source is an array called *CfgByte*, and destination is McBSP1 transmit register located at 0x43; both addresses are in data space. The source address is incremented after every transfer. There are a total of 8 commands, where each command forces the A/D to convert through all 8 channels. Click OK—the DMA configuration object is set. Now to program the *Read* DMA channel.

5. Right click on dmaCfg5 and select properties. In the *General* tab you may choose to add the Comment *read channel* as shown in Figure 19.

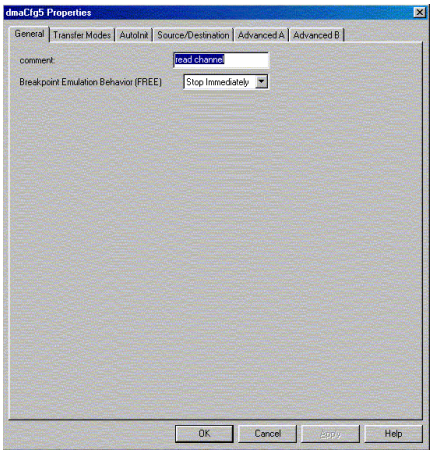

Figure 19. DMA dmacfg5 General Tab

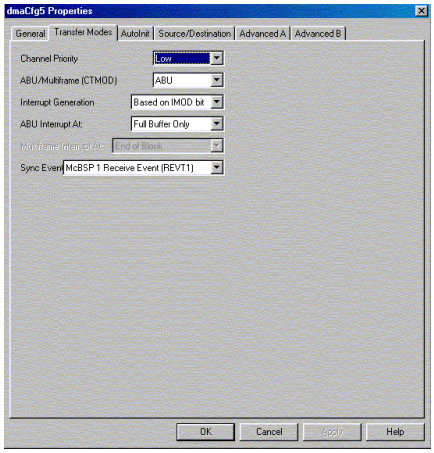

Figure 20. DMA dmaCfg5 Transfer Mode tab

6. In the *Transfer Modes* tab, set options as shown in Figure 20. Select Auto Buffer Mode and synchronize to McBSP receive event. Set Interrupt to occur when buffer is full.

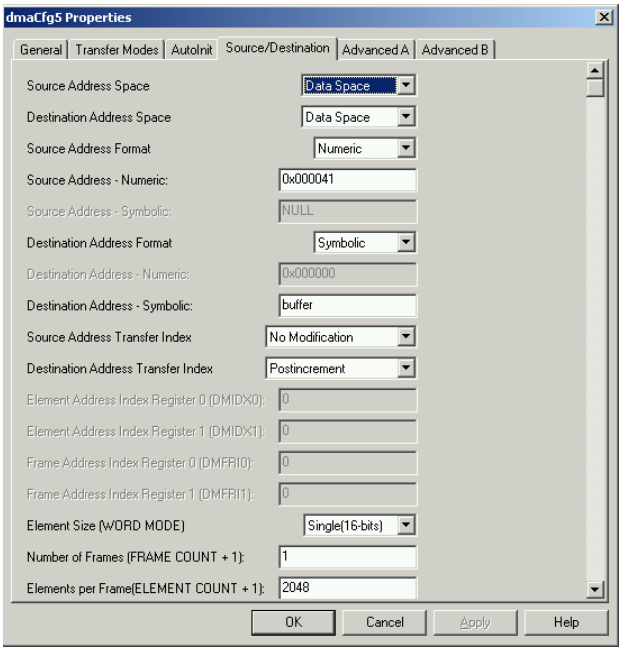

Figure 21. DMA dmaCfg5 Source/Destination Tab

7. In the *Source/Destination* tab, set options as shown in Figure 21. Source is McBSP receive register located at 0x41 in data space. Destination is array called *buffer* also in data space.

The DMA is programmed for auto buffer mode, so the number of transfers is determined by the value written to the 16-bit element register. The buffer size is 2048, therefore set element count register to that value. Now that both DMA configuration objects have been set, they need to be associated with a resource.

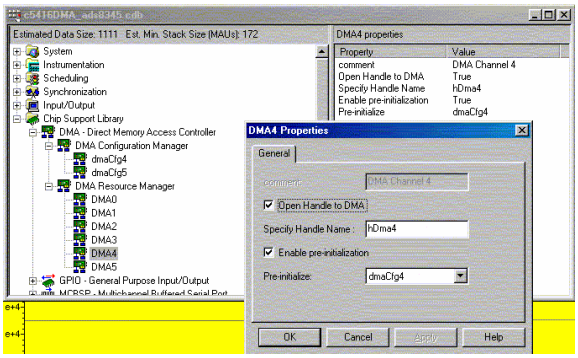

Figure 22. DMA Channel 4 Resource Link

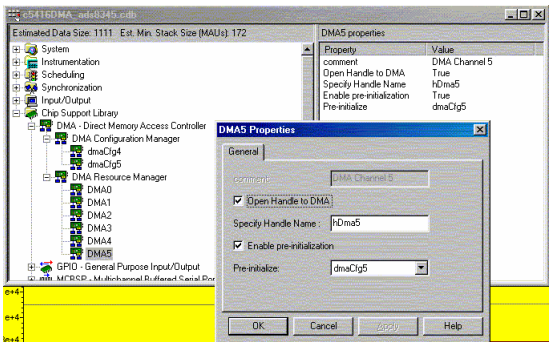

Figure 23. DMA Channel 5 Resource Link

8. Expand the DMA resource Manager directory. Highlight DMA4 and select properties. Check Open Handle and Enable pre-initialization as shown in Figure 22. Select Pre-initialize dmacfg4 from the menu.
9. Highlight DMA5 and select properties. Check Open Handle and Enable pre-initialization as shown in Figure 23. Select Pre-initialize dmacfg5 from the menu.

3.4 HWI Setup

Hardware interrupt service routines can be mapped to interrupt vectors using Configuration Tool.

1. Select and expand *HWI – Interrupt Service Routine Manager* in the scheduling module of DSP/BIOS. Select HWI_SINT13 as shown in Figure 24 and Figure 25.

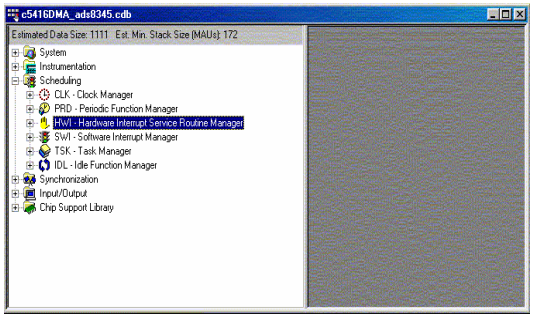

Figure 24. HWI Manager Selection

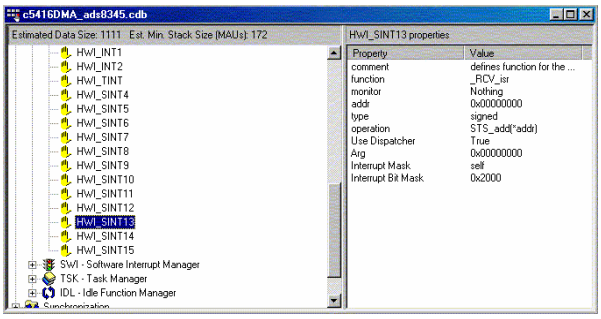

Figure 25. HWI_SNIT13 Selection

2. Right click on HWI_SINT13 and select properties. The *General Tab* is visible; type the name of the ISR to have called when the interrupt occurs. In this application, type *_RCV_isr,* as shown in Figure 26. Whenever a function is typed into the configuration tool, the label must be preceded with an underscore.
3. In the *Dispatcher* Tab, set the options as shown in Figure 27. Enable Dispatcher and interrupt mask self.

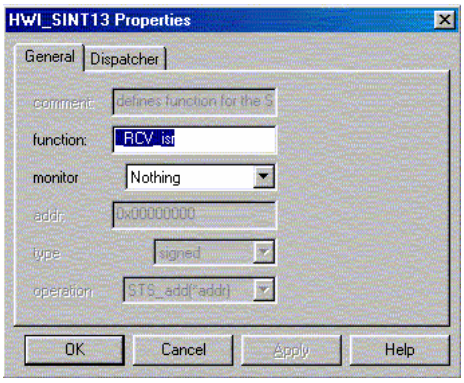

Figure 26. HW SINT13 General Tab

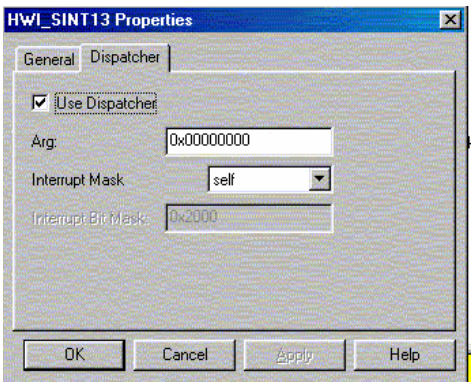

Figure 27. HWI SINT13 Dispatcher Tab

3.5 SWI Setup

Software interrupt (SWI) routines are also mapped using the configuration tool. For each channel there is one function to sort the data, so for each channel we must create 1 SWI object for a total of eight.

1. Expand the *Scheduling Module*. Right click on *SWI-Software Interrupt Manager* and select *insert SWI*. Now rename that object to swiSORTCHAN0, as shown in Figure 28. Repeat the previous Step 7 more times, so the readers' screen is similar to that shown in Figure28.

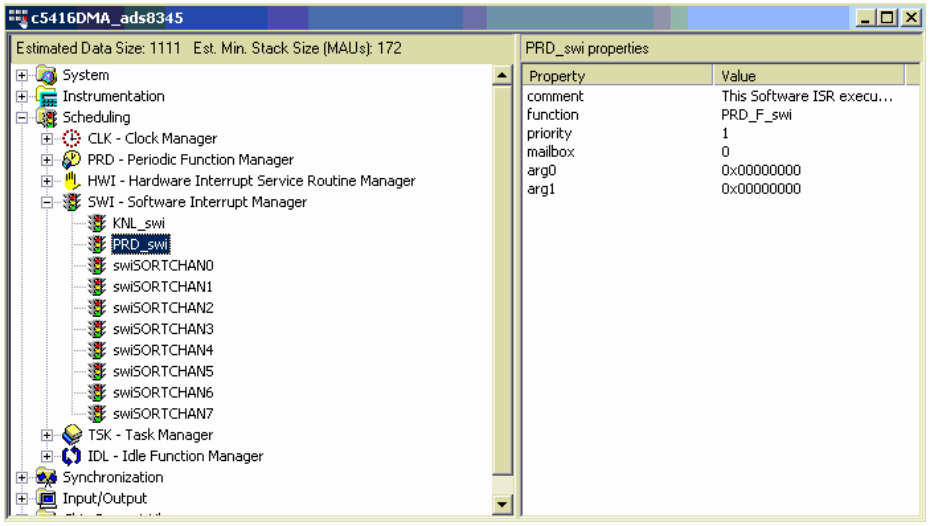

Figure 28 SWI Sort Overview

2. Right Click on *swiSORTCHAN0* and select properties. Change the function property to *_swiSortChan0Func* as shown in Figure 29. Repeat this step for the remaining 7 SWI objects. Change their function property to call the function, which sorts for its respective channel.

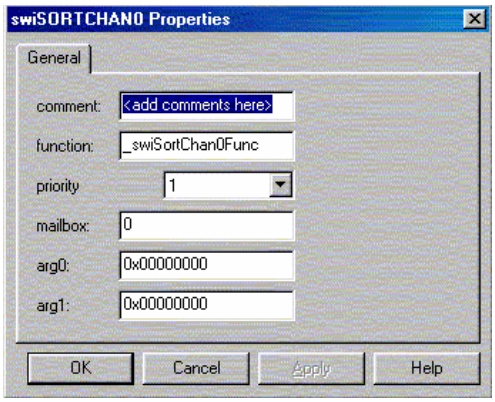

Figure 29. SWI SORTCHAN0 Setup

3.6 Algorithm

In the configuration tool, the pre-initialize option was selected for all the peripherals, meaning the DSP initializes all the peripheral registers during initialization. In the main function, the task then is to enable the peripherals, initialize the analog-to-digital converter, and assert the interrupt. The program written for this application note uses the DMA channels 4 and 5 to read and store 2048 samples (256 from each of the eight channels) from the A/D and then uses the DSP to sort the data.

The analog-to-digital converter does not power up in a known state. So before enabling the DMA to capture the pre-defined samples, the A/D must be initialized in the external clock mode, and single-ended mode. To ensure the A/D is in external clock mode, PD0 and PD1 bits have to be set to high in the first byte command sent after powerup. From then on the A/D remains in external clock mode, and PD0 and PD1 bits can be set to 00 for power down between conversions.

The *ADC_init* function enables the McBSP and writes a configuration byte to the McBSP transmit register. The function ensures the ADS8345 is set for external clock mode and McBSP transmit and receive modules are enabled. Writing 0x0000 to the McBSP 1 transmit register before returning from the function ensures that no additional conversions are enabled until triggered by the DMA channels.

After the *receive* DMA channel stores all 2048 samples, it interrupts the DSP. In the interrupt service routine, all the peripherals are disabled and software interrupts are *posts*, which sort through *buffer* and place each respective channel data into *channel* arrays. For example, digitized data from channel 0 is stored in array *Chan0*, and likewise for all other channels.

Figure 30, lists all the user functions written for this application note. As implied above, there are eight software interrupt routines (SWIs) to move channel data into *channel* arrays. Also mentioned above, is one hardware interrupt (HWI) that is mapped to DMA channel 5, which reads and stores A/D codes. This HWI disables the peripherals and *posts* eight SWI's. The CPU processes the SWIs after exiting the HWI function.

To see the code in action, download the associated zip file and load workspace file *C5416_ADS8345DMA.wks*. Apply analog signals to A/D channels. Load the compiled program file *C5416_ADS8345DMA.out,* and click *Run* from the debug menu.

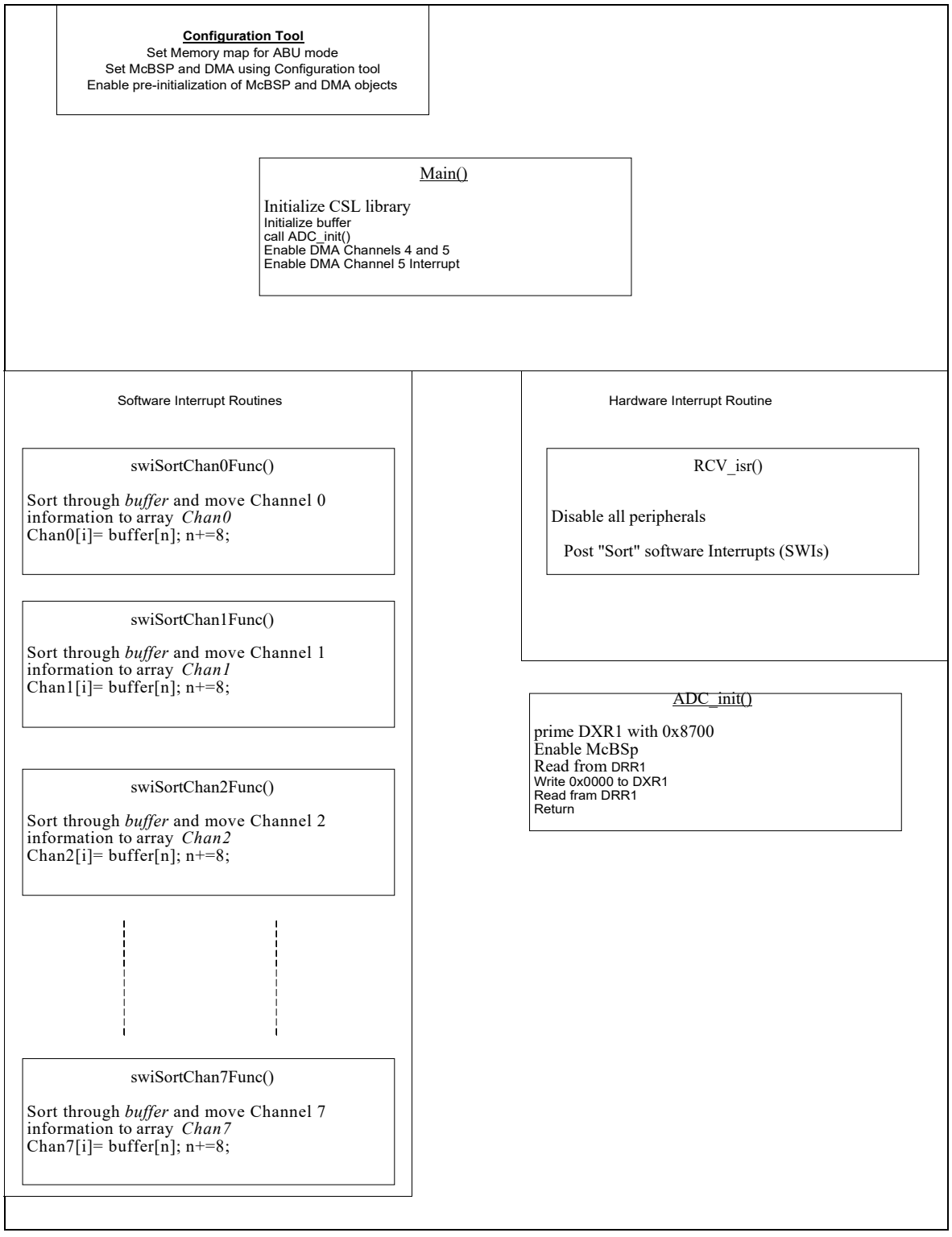

Figure 30. Software Functions

4 Conclusion

This report presents both a hardware and software solution for interfacing the ADS8345 to the C5416™ DSP. It is important to note, this solution can be applied to the ADS8343, ADS8341 and ADS8344, ADS7844 and ADS7841 analog-to-digital converters.

5 References

1. TMS320 DSP/BIOS user's guide (SPRU423)
2. Code Composer Studio Getting Started Guide (Rev. C) (SPRU509)
3. TMS320C54x Optimizing C/C++ Compiler user's guide (SPRU103)
4. TMS320VC5416 data sheet (SPRS095)
5. TMS320C54x DSP Enhanced Peripherals reference set, Vol 5 (SPRU302)
6. TMS320C54x Chip Support Library API reference guide (SPRU420)
7. TMS320C5000 DSP/BIOS Application Programming Interface (API) reference guide (SPRU404)
8. TMS320C54x Code Composer Studio Tutorial Online Help (SPRH134)

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