Application Report
SLAA257–July 2005

Interfacing the ADS7881 to the TMS320C6713 DSP

Lijoy Philipose — Data Acquisition Applications

ABSTRACT

This application report presents a solution to interfacing the ADS7881 12-bit parallel interface converter to the TMS320C6713 DSP. The hardware solution is made up of existing hardware, specifically, the ADS7881EVM, ‘C6713 DSK, and 5-6K Interface Board. The software demonstrates how to use an EDMA ping-pong buffer and Timer1 peripherals to collect data at 4 MSPS. Discussed also are some of the key points to remember when writing this software and modifying it for your application. The software developed is made available for download to involve the user in the discussion and as sample code to use in development. Project collateral discussed in this application report can be downloaded from the following URL: www.ti.com/lit/zip/SLAA257.

Contents

List of Figures

List of Tables

Trademarks

TMS320C6000, C5000, C6000, Code Composer Studio are trademarks of Texas Instruments.

1 Introduction

The ADS7881 converter is a 12-bit, high-speed, SAR-type, parallel interface converter. This analog-to-digital converter (ADC) can sample at 4 MHz. This type of converter is often used in closed-loop systems, multichannel, multiplexed, and high-speed data acquisition systems.

This application report presents a hardware and software solution for interfacing and using the converter with the TMS320C6713 digital signal processor (DSP). The software developed uses the EDMA, in combination with Timer1, to collect 4096 samples. Discussed also are some of the key points to remember when interfacing the ADS7881 to host processors. The software code developed with this report is available for download along with this report.

2 Hardware

The hardware solution involved theTMS320C6713 DSK (C6713 DSK), 5-6K Interface Board, and the ADS7881EVM evaluation module. The hardware used in this report is available for order from Texas Instruments.

2.1 *TMS320C6713 DSK*

The TMS320C6713 DSP Starter Kit (DSK) not only provides an introduction to TMS320C6000™ DSP platform technology, but is powerful enough to use for fast development of networking, communications, imaging, and other applications like data acquisition. For more information, search for part number TMDSDSK6713 at www.ti.com.

2.2 *ADS7881EVM*

The ADS7881 evaluation module is an easy way to test both the functional and dynamic performance of this 12-bit ADC. The evaluation module includes those circuits essential to showing the performance of the converter and interfacing it to a parallel bus. These circuits include the analog input, reference, power, digital buffer circuits, and 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 signal can be applied via standard 0.1-inch IDC header/socket (P1) or via SMA connectors (J2). The buffered data bus is available via 0.1-inch IDC header/socket connectors (P3). The ADS7881 control inputs are also made available via standard 0.1-inch IDC header/socket (J3). The decode logic used to generate the convert start, read, and reset signals are controlled via connector P2. These standard connectors enable the EVM to be plugged into most prototype boards for rapid evaluation. For information on this product, search www.ti.com for the ADS7881EVM. The ADS7881EVM shares the same PWB as the ADS7891EVM, but differs only in the components installed. See the ADS7881/ADS7891EVM user's guide (SLAU150) for more information. Table 1 lists the jumper settings used in this application report. The decode logic looks for three inputs A0, A1, and A2. For this application, DSP address lines A14, A15, and A16 are mapped to A0, A1, and A2, respectively.

Table 1. Jumper Settings for ADS7881EVM

REFERENCE DESIGNATOR	DESCRIPTION	PINS/PADS	
		1–2	2–3
W1	Short U8 pin 14 to Powerdown/Reset signal	Installed	
	Short U8 pin 13 to Powerdown/Reset signal		Not installed
W2	Short U8 pin 12 to $\overline{\text{CONVST}}$ signal	Not installed	
	Short U8 pin 11 to $\overline{\text{CONVST}}$ signal		Not installed
W3	Short U8 pin 10 to $\overline{\text{RD}}$ signal	Not installed	
	Short U8 pin 8 to $\overline{\text{RD}}$ signal		Not installed
W4	Short inverted BUSY to $\overline{\text{INTC}}$	Installed	
	Short BUSY to $\overline{\text{INTC}}$		Not installed
W5	Short +5VD to +BVDD	Installed	
	Short +3.3VD to +BVDD		Not installed

2.3 *5–6K Interface Evaluation Module*

Many data acquisition evaluation modules (EVM) from Texas Instruments are built to 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 ADS7881EVM plugs onto connectors J10 (analog), J17 (data bus), J18 (control bus), and JP5 (power). See TI literature number SLAU104 for more information on the 5-6K Interface Board or search for keyword *5-6K Interface*.

Table 2 lists the jumper settings for the 5-6K Interface Board. Be sure to short across pins 5 and 6 of connector J13 on the 5-6K Interface Board. It routes the BUSY signal from the ADS7881 EVM to INTd on 5-6K Interface Board and finally to external interrupt number seven (INT7) signal to the DSP.

Table 2. Jumper Settings for 5-6K Interface Board

REFERENCE DESIGNATOR	DESCRIPTION	JUMPER	
		PIN 1–2	PIN 2–3
W1	Apply DSP address lines A14 to A17 to ADS7881EVM address lines A0 to A3	OPEN	N/A
W2	Select +5VD from DSK		SHORT
W3	Select +3.3VD from DSK		SHORT
W4	Apply $\overline{\text{CL\_WR}}$	SHORT	
W5	Apply $\overline{\text{DC\_ARE}}$		SHORT
W6	Apply $\overline{\text{DC\_IS}}$		SHORT
W7[1]	Interrupt signal is applied directly to J13.	SHORT	

[1] Revision B of 5-6K Interface Board.

2.4 Hardware Connections

One ADS7881 is mapped into memory space CE2 of the C6713 DSP. The reset and read signal is generated by using a 3-to-8 decoder on the ADS7881EVM. The convert start signal is generated by Timer1. Chip Select and Read signals can be tied LOW. Toggling address lines A16, A15, and A14 generates a reset and read pulses to the ADC. The C6713 has a 32-bit data bus; therefore, the BYTE line is tied LOW, enabling 12-bit bus operation. The inverted BUSY signal is applied to the external interrupt pin 7 of the DSP. If other devices were not on the parallel bus, the RD signal could also be set low. If power consumption were a concern, this scheme would need to be revised for one that allowed the power-down features to be activated. Finally, the data bus is mapped LSB to LSB. The hardware connections are as shown in Figure 1.

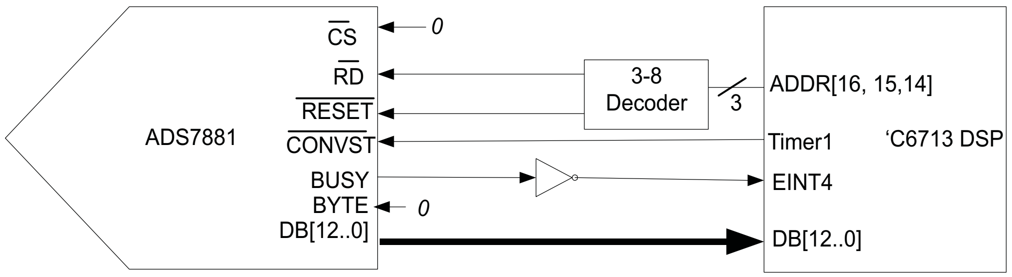


Figure 1. Hardware Connection

3 Software Interface

The software interface objective is to collect a block of samples as quickly as possible, while freeing up the processor to perform more meaningful tasks. The processor should be alerted only when the samples are ready for processing. The most efficient way of doing this is to use an EDMA channel, a timer, and one interrupt routine. For this discussion, it is assumed that the reader is familiar with the DSP and its peripherals. If not, the reader should study the application reports and data sheets listed in the references section at the end of this document.

The Enhanced DMA with the timer is a highly efficient method of gathering data from the ADC (see Figure 2). The timer can be set to a desired frequency and used to trigger each conversion cycle independently. The decoder inputs tied to the address lines are used to create the reset and read pulses. The EDMA channel used in this application is channel 7.

The program is written to collect 4096 continuous samples using a ping-pong buffer scheme. This data capture scheme is possible because the EDMA channel can be linked to different configurations. Exactly how this is done is explained in the next few sections.

The first 2048 sample data words are stored by the EDMA channel into array pingBuf[]. The second set of 2048 samples are stored in array pongBuf[]. After each 2048 samples, the EDMA controller interrupts the DSP. The DSP responds by calling hwiEDMA_isr() function. If the pingpong index is zero, the DSP resets the EDMA controller interrupt registers and returns. If the pingpong index is one, then the CPU disables the Timer1, EDMA channel, and copies all 4096 12-bit data words to buffers dataping[] and datapong[] (see Figure 3).

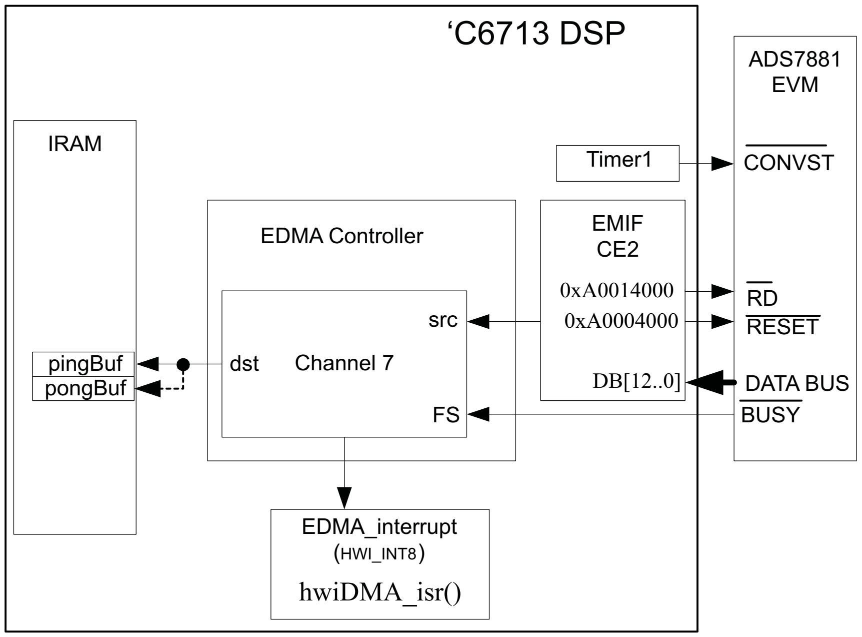


Figure 2. Data Transfer Block Diagram Using EDMA

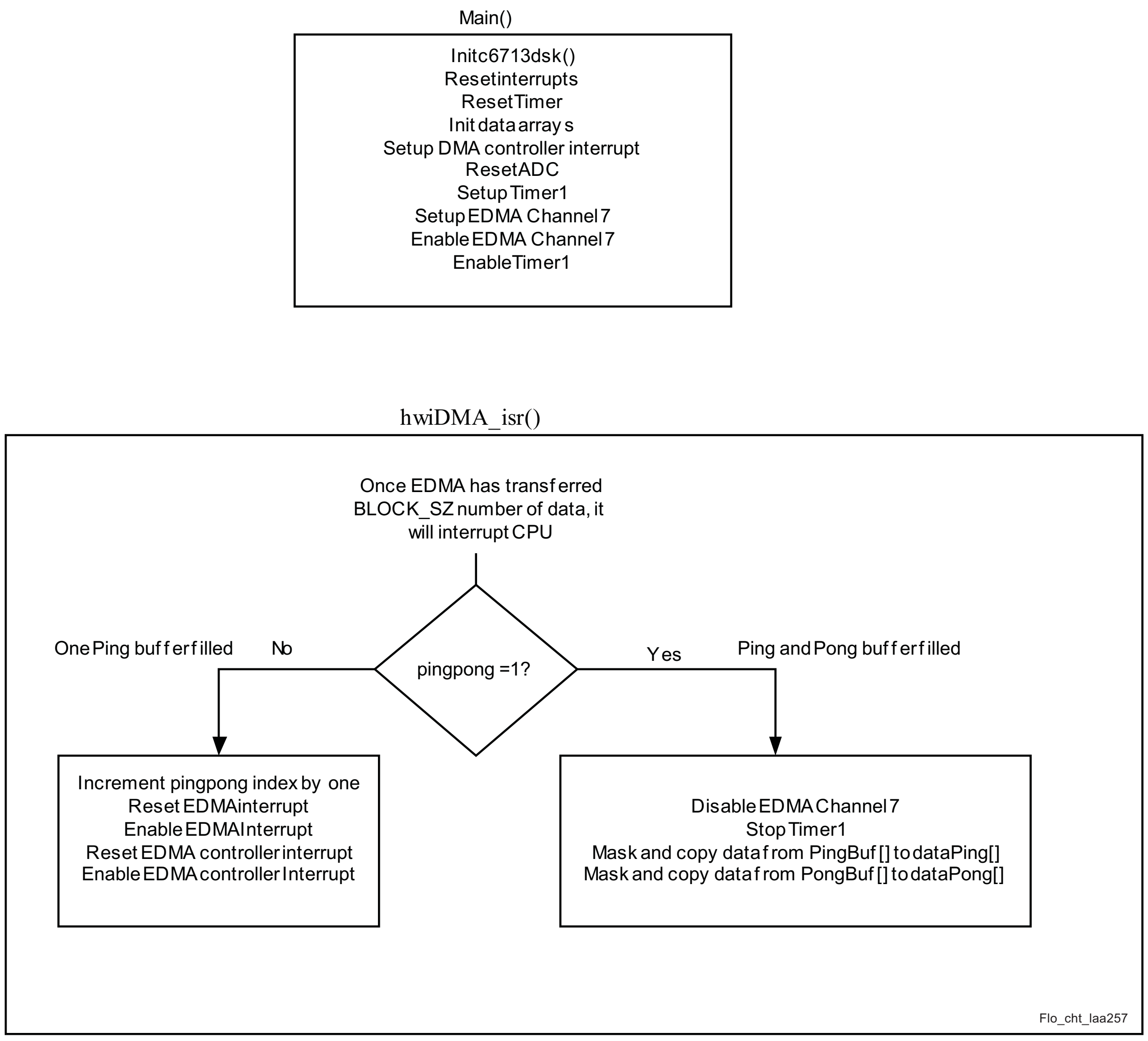

Main()
Initc6713dsk()
Resetinterrupts
ResetTimer
Init data arrays
Setup DMA controller interrupt
ResetADC
Setup Timer1
Setup EDMA Channel 7
Enable EDMA Channel 7
EnableTimer1
hwiDMA_isr()
Once EDMA has transferred BLOCK_SZ number of data, it will interrupt CPU
pingpong =1?
One Ping buffer filled
No
Yes
Ping and Pong buffer filled
Increment pingpong index by one
Reset EDMAinterrupt
Enable EDMAInterrupt
Reset EDMA controller interrupt
Enable EDMA controller Interrupt
Disable EDMA Channel 7
Stop Timer1
Mask and copy data from PingBuf[] to dataPing[]
Mask and copy data from PongBuf[] to dataPong[]
Flo_cht_laa257

Figure 3. Functions Flowchart

3.1 ADS7881

The ADS7881 is a 12-bit, 4-MSPS analog-to-digital converter with a 2.5-V internal reference. It is a successive approximation register (SAR)-type converter. These converters work in two modes – sample mode and hold mode. In sample mode, the converter sample-and-hold capacitor is connected to the input pins of the connector. During this time, the analog input buffer is trying to settle the signal to half an LSB. Assuming a 2.5-V reference and a single-ended input range of 2.5 V, half of an LSB is 305 μV. In hold mode, the sample-and-hold capacitor is disconnected from the input pin. In this mode, the converter is trying to resolve the charge stored on the sample-and-hold capacitor. At the end of this process (when BUSY signal goes HIGH), a digital representation of the charge is ready to be read out by the processor.

The converter features both a 12-bit and an 8-bit interface, along with a pseudo-differential input. The minus input pin can swing ±200 mV, which allows for compensating for ground mismatches between the transducer and converter. It also features a couple of power-down modes: nap mode and power down. Nap mode is always on and is activated when the device is sampling at slower conversion rates. The full power-down feature reduces current drawn from 22 mA to 2.5 μA.

The converter allows users to start sampling in three ways and conversion in two ways. See the ADS7881 data sheet for more information. In this application report, sampling starts at the rising edge of CONVST, as shown in Figure 3 of the PDS.

Figure 4 is a screen shot of the timing captured from connecter J3 of the ADS7881EVM.

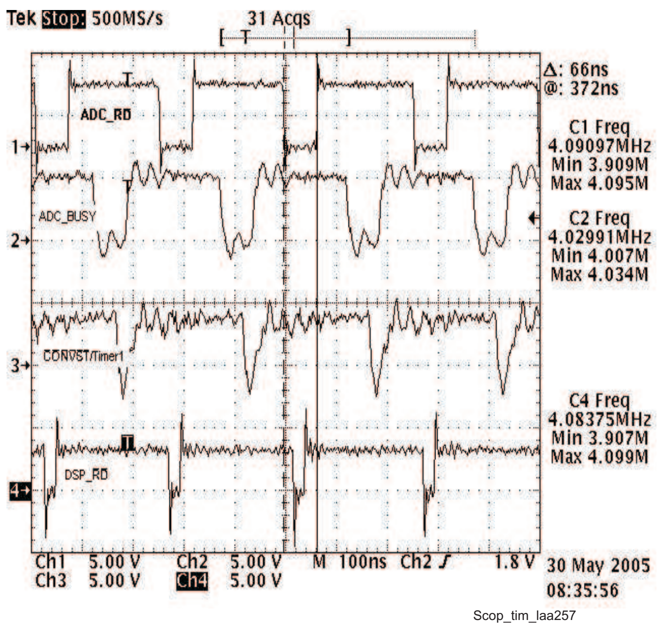
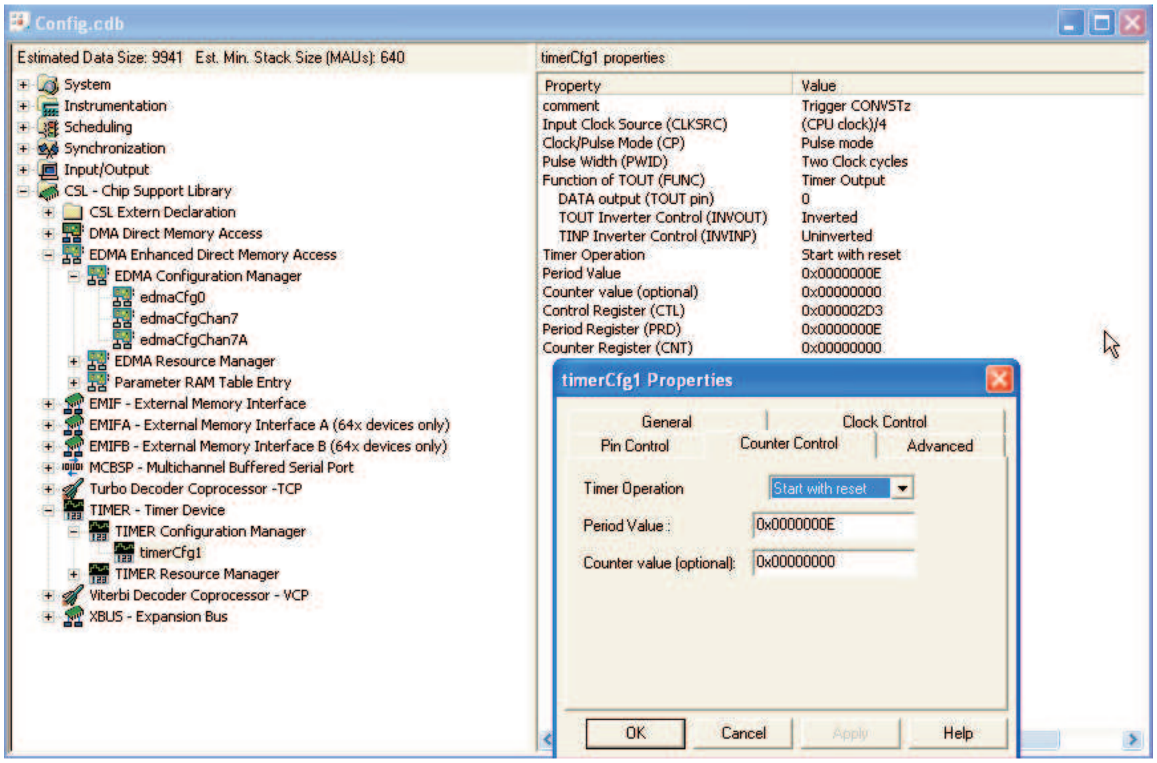


Figure 4. Scope Screenshot of Interface Timing

3.2 C6713 DSP Settings

The TMS320C6713 DSP and its respective peripherals (i.e., Timer1, EMIF and EDMA) need to be set up to work with the converter. It is assumed that the reader has a working understanding of the host processor; therefore, the DSP setup is not covered in much detail in this application report. See the downloadable code and references listed in Section 5 for more information.

The settings for the various peripherals can be found and modified in the config.cdb file, as shown in Figure 5. The seed file for the DSP/BIOS configuration file was the dskC6713.cdb file.



Estimated Data Size: 9941 Est. Min. Stack Size (MAUs): 640

- System
- Instrumentation
- Scheduling
- Synchronization
- Input/Output
- CSL - Chip Support Library
  - CSL Extern Declaration
  - DMA Direct Memory Access
  - EDMA Enhanced Direct Memory Access
    - EDMA Configuration Manager
      - edmaCfg0
      - edmaCfgChan7
      - edmaCfgChan7A
    - EDMA Resource Manager
    - Parameter RAM Table Entry
  - EMIF - External Memory Interface
  - EMIFA - External Memory Interface A (64x devices only)
  - EMIFB - External Memory Interface B (64x devices only)
  - MCBSP - Multichannel Buffered Serial Port
  - Turbo Decoder Coprocessor -TCP
  - TIMER - Timer Device
    - TIMER Configuration Manager
      - timerCfg1
    - TIMER Resource Manager
  - Viterbi Decoder Coprocessor - VCP
  - XBUS - Expansion Bus

timerCfg1 properties

Property	Value
comment	Trigger CONVSTz
Input Clock Source (CLKSRC)	(CPU clock)/4
Clock/Pulse Mode (CP)	Pulse mode
Pulse Width (PWID)	Two Clock cycles
Function of TOUT (FUNC)	Timer Output
DATA output (TOUT pin)	0
TOUT Inverter Control (INVOUT)	Inverted
TINP Inverter Control (INVINP)	Uninverted
Timer Operation	Start with reset
Period Value	0x0000000E
Counter value (optional)	0x00000000
Control Register (CTL)	0x000002D3
Period Register (PRD)	0x0000000E
Counter Register (CNT)	0x00000000

timerCfg1 Properties

Timer Operation: Start with reset
Period Value : 0x0000000E
Counter value (optional): 0x00000000

Scren_cnf_laa257

Figure 5. Screenshot of DSP/BIOS Configuration File

The register settings for EDMA channel 7 follows. Recall that this channel is used to read data from the converter. The source is the address which generates the read pulse using the decoder on the ADS7881EVM. An array called *pingBuf[]* is the destination for the first block of 2048 data points. The destination for the second block of 2048 data is pongBuf. Both arrays are BLOCK_SZ (2048) words long. The EDMA channel 7 transfers NUMSAMPLES(2048) and then loads the second configuration which sets the destination address to the *pongBuf[]* array. After 4096 words are transferred from the ADC, it loads the third configuration which sets the destination address to temp. Each EDMA transfer is synchronized to the external hardware interrupt number 7. In this case, it is the BUSY signal from the ADS7881EVM. EDMA transfers that are synchronized to a rising edge of the frame sync signal. The frame sync signal is the rising edge of the BUSY signal. The rising edge of BUSY indicates the start of a conversion and not the end. Therefore, the first read operation and the first point in the pingBuf[] array should be discarded.

```
EDMA_Config edmaCfgChan7 = { 0x28360003, /* Option */ 0xA0014000, /* Source Address -
 Numeric */ 0x00000000, /* Transfer Counter -
 Numeric */ (Uint32) pingBuf, /* Destination Address -
 Extern Decl. Obj */ 0x00000000, /* Index register -
 Numeric */ 0x00010000 /* Element Count Reload and Link Address */ };
```

```
EDMA_Config edmaCfgChan7A = { 0x28360003, /* Option */ 0xA0014000, /* Source Address -
 Numeric */ 0x00000000, /* Transfer Counter -
 Numeric */ (Uint32) pongBuf, /* Destination Address -
 Extern Decl. Obj */ 0x00000000, /* Index register -
 Numeric */ 0x00010000 /* Element Count Reload and Link Address */ };
```

At higher sampling rate, the DSP is not able to disable the timer and the EDMA channel right away after it collects 4096 samples. As long as the timer and the EDMA channel are enabled, they continue to trigger and store conversion data. The data is stored at the address specified in the EDMA channel 7 destination register. There is a chance the last point in the data array will be overwritten many times before the EDMA channel is disabled. To avoid corruption of sampled data, the EDMA channel is linked to configuration edmaCfg0. After the *read* EDMA channel captures 4096 samples, the EDMA begins storing sample data to the location variable temp. By using the linking feature of the EDMA and the timer, it was possible to easily implement a ping-pong buffer and capture a block number of data samples at 4 MSPS.

```
EDMA_Config edmaCfg0 = { 0x48160003, /* Option */ 0xA0004000, /* Source Address -
 Numeric */ 0x00000000, /* Transfer Counter - Numeric */ (Uint32) &temp, /* Destination Address -
 Extern Decl. Obj */ 0x00010001, /* Index register -
 Numeric */ 0x00010000 /* Element Count Reload and Link Address */ };
```

To see how this linking feature is accomplished, open file Config.cdb in Code Composer Studio™ and expand as shown in Figure 5. The EDMA configuration 7(edmaCfgChan7) tells the EDMA to write 2048 words to pingBuf[]. EDMA configuration 7A(edmaCfgChan7a) forces the EDMA to write 2048 words to pongBuf[]. EDMA configuration 0(edmaCfg0) forces the EDMA to write data read from the converter to variable temp. The EDMA continues to write the samples to temp until the DSP disables Timer1 and the EDMA channel.

The register setting for the Timer1 which sets the sampling rate follows. The Timer1 input clock source is the CPU CLOCK divided by 4.

Timer1 Input Clock = 225E6/4 = 56.25 MHz

The formula for setting the sampling frequency (Fs) is

Fs = 56.25e6 /(Period Value)

Or

Period Value = 56.25e6/(Fs)

```
TIMER_Config timerCfg1 = { 0x000002D3, /* Control Register (CTL) */ 0x0000000E, /* Period
Register (PRD) */ 0x00000000 /* Counter Register (CNT) */ };
```

The Timer1 output is set for clock mode, and the period value is 14. It takes the EDMA about 130 ns after the BUSY rising edge to generate a read pulse (see Figure 4). For this reason, it is possible to use BUSY to trigger conversions and the read at full speed. The disadvantage to this solution is that the first conversion result is invalid because the EDMA reads the converter before the first conversion cycle is complete. Generally, during conversion time, the IC should be kept free of switch currents that could disrupt the ground reference plane of the converter. If this converter is the only one using the data bus, the ADS7881 read pin can be held low.

To change the timer frequency, open file *Config.cdb* in Code Composer Studio and expand as shown in Figure 5. Right-click timerCfg1, select properties, and select counter control tab.

4 Conclusion

This application report presents one solution to interfacing the ADS7881 converter to the C6713 DSP. The ADS7881EVM plugs onto the 5-6K Interface Board, which in turn plugs onto the C6713 DSK. The upgrade device, the ADS7891, is pin-compatible and features 14-bits of resolution and a sampling rate of up to 3 MSPS. All the hardware used for this application report can be ordered from Texas Instruments. The software solution involves using the DSP/BIOS and the configuration tool (i.e., config.cdb file) to visually set up the EDMA, Timer, and interrupts. The EDMA and Timer1 was used to trigger and read conversion data at 4 MHz. The ping-pong buffering scheme employed uses the linking feature of the EDMA. In this particular application, the EDMA halts after collecting 4096 samples. The user can change the program to continuously ping-pong between the two buffers by changing the link handle in edmaCfgChan7A to hEdmaTbl7, instead of hEdmaTbl0.

5 References

1. *TMS320C621x/TMS320C671x EDMA Architecture* application report (SPRA996)
2. *Applications Using the TMS320C6000 Enhanced DMA* application report (SPRA636)
3. *TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) Controller Reference Guide* (SPRU234)
4. *ADS7881, 12-Bit, 4-MSPS Low Power SAR Analog-To-Digital Converter* data sheet (SLAS400)
5. TMS320C6713, TMS320C6713B, Floating-Point Digital Signal Processor data sheet (SPRS186)
6. *ADS7881 /ADS7891EVM User's Guide* (SLAU150)
7. *5-6K Interface Board EVM User's Guide* (SLAU104)
8. *TMS320C67x/C67x+ DSP CPU and Instruction Set Reference Guide* (SPRU733)
9. *TMS320C6000 DSP External Memory Interface (EMIF) Reference Guide* (SPRU266)

10. *TMS320C6000 DSP 32-Bit Timer Reference Guide* (SPRU582)

MAIN.C

```
/*****************************************************************/ /* File: main.c */ /*
Description: Program for interfacing ADS7881 */ /* to an C6713 DSK. Program uses the Timer1 to
trigger a */ /* convst# pulse, at about 4MHz. */ /* Inverted BUSY signal is used to trigger EDMA
channel 7 to */ /* read from A/D and store data into ad_buffer. Once BLOCK_SZ */ /* of data is
captured, the EDMA will interrupt CPU. CPU will */ /* then halt EDMA channels and Timer1. */ /*
*/ /* AD converter address: CE2 memory space */ /* 0xA0004000 (RESET#) */ /* 0xA0014000 (RD#) */
/* EVM Hardware Connections: */ /* CS# => short J3 pins 1 and 2 */ /* RD# => Generated from 3-
8 decoder mapped to 0xA0004000 */ /* CONVST# => Generated by shorting Timer1 output to W2 pin 2
*/ /* RESET# => Generated from 3-
8 decoder mapped at 0xA0014000 */ /* BUSY => Inverted then wired to EXTERNAL INT7 */ /* AUTHOR :
DAP Application Group, L. Philipose, Dallas */ /* CREATED 2005 © BY TEXAS INSTRUMENTS
INCORPORATED. */ /* VERSION: 1.0 */
/*****************************************************************/

/* Include Header File */ #include "Configcfg.h" #include "dc_conf.h"

/* include files for DSP/BIOS */ #include <std.h> #include <swi.h> #include <log.h>

/* include files for chip support library */ #include <csl.h> #include <csl_legacy.h> #include
<csl_irq.h> #include <csl_timer.h> #include <csl_edma.h>

/* function prototypes */ void init_dsk(void);

/* Create the buffers. We want to align the buffers to be cache friendly */ /* by aligning them
on an L2 cache line boundary. */ #pragma DATA_ALIGN(pingBuf,BLOCK_SZ); #pragma
DATA_ALIGN(pongBuf,BLOCK_SZ); #pragma DATA_ALIGN(dataPing,BLOCK_SZ); #pragma
DATA_ALIGN(dataPong,BLOCK_SZ); unsigned short pingBuf[BLOCK_SZ]; /*data from AD, written to by
EDMA */ unsigned short pongBuf[BLOCK_SZ]; /*data from AD, written to by EDMA */ unsigned short
dataPing[BLOCK_SZ]; /*data from AD, written to by EDMA */ unsigned short dataPong[BLOCK_SZ];
/*data from AD, written to by EDMA */ signed short temp, n=0;

void main(void) { int I;

 /* initialize the EMIF*/ init_dsk(); IRQ_reset(IRQ_EVT_EDMAINT); /*Reset EDMA interrupt */\
IRQ_clear(IRQ_EVT_EXTINT7); TIMER_reset(hTimer1); for (I=0; i<=BLOCK_SZ; i++) { /*Initialize data
buffers */ pingBuf[i]=0x00000; dataPing[i]=0x00000; pongBuf[i]=0x00000; dataPong[i]=0x00000; }

/* Enable the EDMA controller interrupt */ IRQ_enable(IRQ_EVT_EDMAINT);
EDMA_intDisable(TCCINTNUM6); /*Disable EDMA interrupt */ EDMA_intClear(TCCINTNUM6); /*Clear EDMA
interrupt */ EDMA_intEnable(TCCINTNUM6); /*Enable EDMA interrupt */

temp = *ADS7881_RESETZ; /*Reset ADC*/

/*Configuration of Timer1 and EDMA channel 7 */ EDMA_config(hEdmaCha7,&edmaCfgChan7);
TIMER_config(hTimer1,&timerCfg1); EDMA_clearChannel(hEdmaCha7); /*Only Timer1 and Edma Channel
7*/ EDMA_enableChannel(hEdmaCha7); /*Enable EDMA channel 7 -
RD# */ TIMER_start(hTimer1); /*Start A/D CONVST# trigger */

// LOG_printf(&trace, "GO! \n"); /*Go sample at 4MSPS */

/*****************************************************************/ /* end main.c */
/*****************************************************************/
```

Functions.C

```
/*****************************************************************/
/*****************************************************************/ /* File: functions.c */ /*
Description: Functions for interfacing ADS7881 to C6713 */ /* init_dsk, hwiDMA_isr,
swiEnablePrephFunc */ /* AD converter address: CE2 memory space */ /* 0xA0014000 (RD#) */ /*
0xA0004000 (RESET#) */ /* Hardware Connections: */ /* CS# => LOW */ /* RD# => Generated from 3-
8 decoder mapped to 0xA0014000 */ /* CONVST# => Generated is Timer1*/ /* RESET# => Generated from
3-
8 decoder mapped at 0xA0004000 */ /* BUSY => Inverted then wired to EXTERNAL INT7 */ /* AUTHOR :
DAP Application Group, L. Philipose, Dallas */ /* CREATED 2005 © BY TEXAS INSTRUMENTS
INCORPORATED. */ /* VERSION: 1.0 */
/*****************************************************************/ /* Include Header File */
#include "Configcfg.h" #include "dc_conf.h" #include <csl_legacy.h> int pingpong=0; /*=1
pingbuffer full*/ extern unsigned short ad_data[BLOCK_SZ]; extern unsigned short
pingBuf[BLOCK_SZ]; extern unsigned short pongBuf[BLOCK_SZ]; extern unsigned short
dataPing[BLOCK_SZ]; extern unsigned short dataPong[BLOCK_SZ];

/*****************************************************************/ /* init_dsk() */ /* This
initializes the EMIF */ /*****************************************************************/ void
init_dsk(void) {

 UINT32 gblctl,ce0ctl,ce1ctl,ce2ctl,ce3ctl,sdctl,sdtim,sdext; /* intialization of the EMIF */

 /* RBTR8,SSCRT,CLK2EN,CLK1EN,SSCEN,SDCEN,NOHOLD */ gblctl = EMIF_MK_GBLCTL( 0, 0, 1, 0, 0, 0, 0);

 /* RDHLD,MTYPE,RDSTRB,TA,RDSETUP,WRHLD,WRSTRB,WRSETUP */ ce0ctl = EMIF_MK_CECTL( 0, 3, 0, 0, 0,
0, 0, 0);

 /* RDHLD,MTYPE,RDSTRB,TA,RDSETUP,WRHLD,WRSTRB,WRSETUP */ ce1ctl = EMIF_MK_CECTL( 0, 2, 0, 0, 0,
0, 0, 0);

 /* RDHLD,MTYPE,RDSTRB,TA,RDSETUP,WRHLD,WRSTRB,WRSETUP */ ce3ctl = EMIF_MK_CECTL( 0, 2, 0, 0, 0,
0, 0, 0);

 /* TRC,TRP,TRCD,INIT,RFEN,SDWID,SDCSZ,SDRSZ,SDBSZ */ sdctl = EMIF_MK_SDCTL( 7, 1, 1, 1, 1, 0, 1,
0, 0);

 /* PERIOD,XRFR */ sdtim = EMIF_MK_SDTIM( 1562, 0);

 sdext = EMIF_SDEXT_NA;

 /* make CE2 control register value */ /* This is the CE space used by the ADS8402 EVM. */ /* Use
the timing values from dc_conf.h: */ ce2ctl = EMIF_MK_CECTL( EMIF_CECTL_RDHLD_OF (RDHLD), /* read
hold */ EMIF_CECTL_MTYPE_ASYNC32, EMIF_CECTL_RDSTRB_OF (RDSTRB), /* read strobe */
EMIF_CECTL_TA_NA, EMIF_CECTL_RDSETUP_OF (RDSETUP), /* read setup */ EMIF_CECTL_WRHLD_OF (WRHLD),
/* write hold */ EMIF_CECTL_WRSTRB_OF (WRSTRB), /* write strobe */ EMIF_CECTL_WRSETUP_OF
(WRSETUP) /* write setup */ );

 /* configure the EMIF */ EMIF_ConfigB(gblctl,ce0ctl,ce1ctl,ce2ctl, ce3ctl,sdctl,sdtim,sdext);

 return; } /* end init_dsk() */

/*************************************************************/ /*hwiDMA_isr(): */ /* Hardware
Interrupt Function disables EDMA channels and */ /* Timer0, Then post software interrupt. */
/*************************************************************/ void hwiDMA_isr() { int i;

if (pingpong==1){ EDMA_disableChannel(hEdmaCha7); /*Disable EMDA channel 7 -
RD#*/ TIMER_pause(hTimer1); for(i=0;i<BLOCK_SZ;i++){ dataPing[i]=(pingBuf[i] & 0x3FFC)>>2;
dataPong[i]=(pongBuf[i] & 0x3FFC)>>2; //ADS7881 12-bit word }

} pingpong++;

 IRQ_reset(IRQ_EVT_EDMAINT); /*Reset EDMA interrupt */ IRQ_enable(IRQ_EVT_EDMAINT);
EDMA_intDisable(TCCINTNUM6); /*Disable EDMA interrupt */ EDMA_intClear(TCCINTNUM6); /*Clear EDMA
interrupt */ EDMA_intEnable(TCCINTNUM6); /*Enable EDMA interrupt */

/*****************************************************************/ /* End Functions.c */
```

```
/*****************************************************************/
```