Interfacing the ADS8371 to TMS3206713 DSP

Lijoy Philipose

ABSTRACT
This application report presents a solution to interfacing the ADS8371 16-bit, 750-KSPS, parallel interface converter to the TMS320C6713 digital signal processor of the TMS320™ DSP family. The hardware solution is made up of existing hardware, specifically the ADS8371EVM, 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 740 kHz. Some key points to remember when writing this software, and modifying it for your application, are also discussed. The software developed is made available for download to involve the user in the discussion and as sample code to use in application development.

Contents
1 Introduction ................................................................................................................. 1
2 Software Interface ................................................................................................. 4
3 Conclusion ............................................................................................................... 9
4 References .............................................................................................................. 9
Appendix A — MAIN.C .............................................................................................. 10
Appendix B Functions.C ............................................................................................. 12

List of Figures
1 Hardware Connection ............................................................................................... 4
2 Data Transfer Block Diagram Using EDMA ............................................................. 4
3 Screenshot of DSP/BIOS Configuration File ............................................................ 5
4 Read Pulse Delay ...................................................................................................... 8
5 Convert Start Jitter .................................................................................................... 8
6 Full Cycle Timing ...................................................................................................... 9

List of Tables
1 Jumper Settings for EVM ......................................................................................... 2
2 5-6K Interface Board Rev A ..................................................................................... 3
3 5-6K Interface Board Rev B ..................................................................................... 3

1 Introduction
The ADS8371 converter is 16-bit, 750-kHz, parallel interface analog-to-digital converter. This application report presents a hardware and software solution to interfacing and using this converter with the TMS320C6713 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 converters to host processors.
1.1 **Hardware**

The hardware solution involves the TMS320C6713 DSK (C6713 DSK), 5-6K Interface Board, and the ADS8371EVM. The hardware used in this report is available for order from Texas Instruments.

1.2 **TMS320C6713 DSK**

The TMS320C6713 DSP Starter Kit (DSK) not only provides an introduction to C6000™ 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 http://www.ti.com.

1.3 **ADS8371 EVM**

The ADS8371 Evaluation Module (EVM) is an easy way to test both the functional and dynamic performance of this 16-bit analog-to-digital converter. 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 connector (J1). The buffered data bus is available via 0.1-inch IDC header/socket connectors (P3). The ADS8371 control inputs are also made available via standard 0.1-inch IDC header/socket (J2). The decode logic used to generate the convert start, read, and reset signals is 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 http://www.ti.com for the ADS8371EVM. 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>
<thead>
<tr>
<th>Designator</th>
<th>Description</th>
<th>Jumper</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>Apply +5VD to +BVDD</td>
<td>Installed</td>
</tr>
<tr>
<td></td>
<td>Connect +3.3VD from connector to +BVDD</td>
<td>Not installed</td>
</tr>
<tr>
<td>W2</td>
<td>Set A[2..0]= 0x1 to generate RD pulse</td>
<td>Installed</td>
</tr>
<tr>
<td></td>
<td>Set A[2..0]=0x2 to generate RD pulse</td>
<td>Not installed</td>
</tr>
<tr>
<td>W3</td>
<td>Set A[2..0]=0x3 to generate convst pulse</td>
<td>Installed</td>
</tr>
<tr>
<td></td>
<td>Set A[2..0]=0x4 to generate convst pulse</td>
<td>Not installed</td>
</tr>
<tr>
<td>W4</td>
<td>Apply inverted BUSY signal to P2 pin19 (INTC).</td>
<td>Not installed</td>
</tr>
<tr>
<td></td>
<td>Set BUSY pulse to INTc</td>
<td>Installed</td>
</tr>
</tbody>
</table>

1.4 **5-6K Interface Evaluation Module**

Many data acquisition evaluation modules (EVM), from Texas Instruments, are being 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™ DSP and C6000™ DSP Starter Kits.

The 5-6K Interface Board consists of two serial connectors, two signal conditioning areas, and parallel interface. The EVM 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.

Tables 2 and 3 list the jumper settings for the 5-6K Interface Board. Be sure to short across pins 7 and 8 of connector J13 on the 5-6K Interface Board. It routes the INTc from the ADS8371EVM to INTd on 5-6K board, and finally to external interrupt number seven (INT7) pin on the DSP.
1.5 **Hardware Connections**

One ADS8371 is mapped into memory space CE2 of the C6713 DSP. The read-and-convert start signals are generated by using a 3-to-8 decoder on the EVM. Address lines A16, A15, and A14 are used to generate read-and-convert start signals to the analog-to-digital converter. The C6713 has a 32-bit data bus; therefore, the BYTE line is tied LOW enabling 16-bit bus operation. The BUSY signal is applied to the external interrupt pin 7 of the DSP. The chip select (CS) signal is grounded. If power consumption is not a concern in your application, CS can be tied low permanently. The user only needs to provide the read (RD)-and-convert start (CONVST) signals.

Finally, the data bus is mapped LSB to LSB. The hardware connections are shown in Figure 1.
2 Software Interface

The objective is to collect a block of samples as quickly as possible, while freeing up the processor to perform other tasks. The processor should be alerted only when the samples are ready for processing. The most efficient way of doing this is to use a couple of EDMA channels, a timer, and interrupts. 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.

It takes the DSP a fixed time to accomplish this task. The enabled timer continues to trigger new conversions, and the EDMA continues to start new conversions and read data from the converter. At higher sampling rates, it takes the CPU a number of conversion cycles before it disables the timer and EDMA channels. The EDMA reads data from the converter and stores it at the destination address. For this reason, it is recommended the EDMA channel be linked to transfer data to another location after it has completed collecting the desired number of samples. To see how this can be accomplished, open file Config.cdb in Code Composer Studio™ and expand as shown in Figure 3. 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 from the converter to variable temp. The EDMA continues to write the samples to temp until the CPU disables Timer1, and the respective EDMA channels.
2.1 ADS8371

The ADS8371 is a successive approximation register (SAR)-type converter. SAR-type converters work in two modes, sample and hold. In sample mode, the converter sample-and-hold capacitor is connected to the input pins of the connector. In this mode, the analog input buffer must settle the signal to half LSB. Assuming a 4.096-V reference and a single-ended input range of 4.096-V, half of an LSB is 62.5 µV. In hold mode, the sample-and-hold capacitor is disconnected from the input pins. When the converter enters the hold mode, the voltage difference between the +IN and –IN inputs is captured on the internal capacitor array.

The voltage on the –IN input is limited between –0.2 V and 0.2 V, allowing the input to reject small signals which are common to both the +IN and –IN inputs. The +IN input has a range of –0.2 V to Vref + 0.2 V. The input span (+IN – –IN) is limited to 0 V to Vref.

The ADS8371 can operate with an external reference (Vref) source ranging from 2.5 V to 4.2 V. The reference voltage on the input pin 1 (REFIN) of the converter is internally buffered. A clean, low-noise, well-decoupled reference voltage on this pin is required to ensure good performance of the converter.

The first three conversions after power-up are used to load factory-stored trim data for the specific device to ensure the specified accuracy. The results of the first three conversions are invalid and should be discarded.

The device can be reset by combinations of CS and CONVST while the BUSY signal is high. Firstly, a CONVST is issued when CS is low and internal CONVERT state is high. The falling edge of CONVST triggers the reset. Secondly, a CS is issued while internal CONVERT state is high. The falling edge of CS triggers the reset. Either one of the before-mentioned conditions triggers an internal self-clear reset to the converter. Once the device is reset, all output latches are cleared (data bus set to zero) and the BUSY signal is brought low. A new sampling period starts at the falling edge of the BUSY signal immediately after the instant of the internal reset.
Software Interface

Certain precautions are necessary to ensure that transitions on the digital inputs do not affect converter performance. The converter switches from sample to hold mode on the falling edge of the CONVST input pin. A clean and low-jitter falling edge is important to achieve the best performance from the converter. A sharp falling transition on the CONVST pin can affect the voltage stored on the sample-and-hold capacitor. A falling transition time in the range of 10 ns to 30 ns is required to achieve the rated performance of the converter. All the signals in the screen shots shown in this report were taken with the oscilloscope probe at connector J2 of the ADS8371EVM. The actual transition times of RD and CONVST at the converter pins are similar to that of the BUSY signal shown in the screen shots.

Also, to ensure best performance, it is recommended that all activity on the input pins happen during the first 400 ns of the conversion period and 125 ns before falling edge of CONVST.

2.2 C6713 DSP

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 detail in this paper. See the downloadable code and references listed in Section 4 for more information.

The settings for the various peripherals can be found and modified in the config.cdb file, as shown in Figure 3. The seed file for the DSP/BIOS configuration file was the dskC6713.cdb file.

The following code listing shows the registers' settings for EDMA channel 2 which is used to trigger a conversion cycle. The source is the address that generates the convert start pulse through the decoder on the EVM. The destination for the data transfer is a variable called temp—used to store data that can be discarded. The EDMA should transfer NUMSAMPLES, or 2048 sample data points. NUMSAMPLES is defined in dc_conf.h file. Each transfer frame is synchronized to a Timer1 event. As a result, the frequency of each transfer, or convert start pulse, is the frequency of the Timer1.

```c
EDMA_Config edmaCfgChan2 = {
    0x28020003, /* Option */
    0xA000C000, /* Source Address - Numeric */
    0x00000000, /* Transfer Counter - Numeric */
    (UInt32) &temp /* Destination Address - Extern Decl. Obj */
    0x00000000, /* Index register - Numeric */
    0x00010000 /* Element Count Reload and Link Address */
};
```

The following code listing shows the register settings for EDMA channel 7. 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 EVM. An array called pongBuf[] 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 A/D converter, it loads the third configuration which sets the destination address to the address of temp. Each EDMA transfer is synchronized to the external interrupt 7. In this case, it is the BUSY signal from the ADS8371EVM. EDMA transfers 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 a start of a conversion and not the end; therefore, the first point in the pongBuf[] array should be discarded.

```c
EDMA_Config edmaCfgChan7 = {
    0x28360000, /* Option */
    0xA0004000, /* Source Address - Numeric */
    0x00000000, /* Transfer Counter - Numeric */
    (UInt32) pongBuf, /* Destination Address - Extern Decl. Obj */
    0x00010001, /* Index register - Numeric */
    0x00010000 /* Element Count Reload and Link Address */
};
```
EDMA_Config edmaCfgChan7A = {
    0x28360003, /* Option */
    0xA0004000 /* Source Address - Numeric */
    0x00000000, /* Transfer Counter - Numeric */
    (Uint32) pongBuf, /* Destination Address - Extern Decl. Obj */
    0x00010001, /* Index register - Numeric */
    0x00010000 /* Element Count Reload and Link Address */
};

At high sampling rates, the DSP cannot disable the timer and the EDMA channels right away after collecting all the required samples. So long as the timer and the EDMA channels are enabled, they continue to trigger and transfer data. The data is stored at the address specified in EDMA channel 7 destination register. So, there is a chance the last point in the data array is overwritten many times before the EDMA channel is disabled in the hardware interrupt service routine. To avoid corruption of the 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 variable temp. By using the linking feature of the EDMA, it is possible to easily implement a ping-pong buffer and find a way to capture a number of samples at 740 KSPS.

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 */
};

The register setting for Timer1 which sets the sampling rate is shown below. 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 ($F_s$) is:

$$F_s = \frac{56.25e6}{2\times \text{Period Value}} \text{ OR}$$

Period Value = $\frac{56.25e6}{2\times F_s}$

TIMER_Config timerCfg1 = {
    0x000000D1, /* Control Register (CTL) */
    0x00000026, /* Period Register (PRD) */
    0x00000000 /* Counter Register (CNT) */
};

The Timer1 output is set for clock mode and the period value is set to 38d.

This particular converter is not as susceptible to minor changes to the ground reference during the first 400 ns after start of conversion. As a general rule during conversion time, the IC should be kept free of switch currents that could disrupt the ground reference plane of the converter.
The ADS8371 requires a quiet zone of 125 ns before the CONVST start falling edge. It also requires all I/O operations to occur during the first 400 ns after conversion starts. It takes the EDMA 120 ns to 132 ns to generate a pulse, once triggered (see Figure 4). Triggering the EDMA read converter channel to the rising edge of BUSY ensures that these timing constraints are met. Unfortunately, this scheme of reading the converter during a conversion cycle causes the first data point to be invalid.

Inherent in this interface scheme is that it has about 10-ns to 12-ns of jitter on the convert start pulse by the fact that it is generated by the EDMA. Figure 5 shows the Timer1 frequency is at 740 kHz, but the convert start-pulse frequency varies from 738 kHz to 744 kHz cycle-to-cycle. In applications where this level of jitter is unacceptable, the user may choose to short the timer output directly to CONVST pin of the converter. The Timer1 needs to be reprogrammed for inverted pulse mode, two clocks wide.

To change the timer frequency, open file Config.cdb in Code Composer Studio and expand as shown in Figure 3. Right click timerCfg1, select properties, and select counter control tab.
3 Conclusion

This application report presents one solution to interface the ADS8371 converter to the C6713 DSP. The ADS8371EVM plugs onto the 5-6K Interface Bard, which in turn plugs onto the C6713 DSK. All the hardware used for this application report can be ordered from Texas Instruments. The software solution uses the DSP/BIOS and configuration tool (i.e., config.cdb file) to visually set up the EDMA, timer, and interrupts. The EDMA and Timer1 were used to trigger conversion cycles at about 740 kHz. The ping-pong buffering scheme was employed using the linking feature of the EDMA. In this particular application, the EDMA halts after collecting 4096 samples. The user may change the program to continuously ping pong between the two buffers by changing the link handle in edmaCfgChan7A to hEdmaTbl7, instead of hEdmaTbl0.

4 References

1. TMS320C621x/TMS320C671x EDMA Architecture (SPRA996)
2. TMS320C6000 Enhanced DMA: Example Applications (SPRA636)
4. ADS8371, 16-Bit, Unipolar Input, Micro Power Sampling, Analog-to-Digital Converter with Parallel Interfacing data sheet (SLAS390)
5. ADS8411, 16-Bit, 2 MSPS, Unipolar Input, Micro Power Sampling, Analog-to-Digital Converter with Parallel Interface and Reference data sheet (SLAS369)
6. TMS320C6713, Floating-point Digital Processors, data sheet (SPRS186)
7. ADS8371/EVM User's Guide (SLAU137)
8. 5-6 K Interface Board EVM User's Guide (SLAU104)
/** File: main.c */
/*
* Description: Program for interfacing ADS8371 to a 'C6713 DSK. Program uses the Timer1 to trigger EDMA a transfer which causes a convst# pulse, at about 740 kHz. The BUSY signal is used to trigger EDMA channel 7 to read from the A/D and store data into pingBuf[] and pongBuf[] data arrays. 
* Once 2048 of data is captured, the EDMA will interrupt CPU. CPU will then halt EDMA channels and timer1 after the second time. 
* The program collects 4096 samples. EDMA triggers of the rising edge of BUSY. 
* AD converter address: CE2 memory space 
* 0xA0004000 (RD#) 
* 0xA000C000 (CONVST#) 
* Hardware Connections: 
* CS# => Shorted to Ground 
* RD# => Generated from 3-8 decoder mapped to 0xA0004000 
* CONVST# => Generated from 3-8 decoder mapped at 0xA000C000 
* BUSY => EXTERNAL INT7 
* AUTHOR : DAP Application Group, L. Philipose, Dallas 
* CREATED 2004(C) BY TEXAS INSTRUMENTS INCORPORATED. 
* VERSION: 1.0 
*/
*********************************************************************/

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

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

/* 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); /*ping buffer*/
#pragma DATA_ALIGN(pongBuf,BLOCK_SZ); /*pong buffer*/
unsigned short pingBuf[BLOCK_SZ];
unsigned short pongBuf[BLOCK_SZ];
unsigned short temp, n=0;

void main(void)
{
    int i;

    /* initialize the EMIF and A/D*/
    init_dsk();
    init_adc();
    /*Initialize data buffers */
    for (i=0; i<=BLOCK_SZ; i++) {
        pingBuf[i]=0x0000;
        pongBuf[i]=0x0000;
    }

    /* Enable the EDMA controller interrupt */
    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 */

/*Configure EDMA Channels, clear and enable them */
EDMA_config(hEdmaCha7, &edmaCfgChan7);
EDMA_config(hEdmaCha2, &edmaCfgChan2);
EDMA_clearChannel(hEdmaCha2);
EDMA_clearChannel(hEdmaCha7);
EDMA_enableChannel(hEdmaCha2);  /*Enable EDMA channel 2 - CONVST# */
EDMA_enableChannel(hEdmaCha7);   /*Enable EDMA channel 7 - RD# */

/*Timer1 was initialized by DSP/BIOS before entering main.c */
/*Only need to enable it here. */
  TIMER_start(hTimer1);  /*Start A/D CONVST# trigger */

/*Go sample at 740kSPS*/
}  
/*end main.c */
/********************************************************************************
/*----------------------------------------------------------------------------*/
Appendix B

Appendix B  Functions.C

/******************************************************************************
* File: functions.c *
* Description: Functions for interfacing ADS8371 to C6713 *
* init_dsk(), init_adc(), hwIDMA_isr(), swIEnablePrephFunc() *
* AD converter address: CE2 memory space *
* 0xA0004000 (RD#) *
* 0xA000C000 (CONVST#) *
* Hardware Connections: *
* CS# => Grounded *
* RD# => Generated from 3-8 decoder mapped to 0xA0004000 *
* CONVST# => Generated from 3-8 decoder mapped at 0xA000C000 *
* BUSY => EXTERNAL INT7 *
* AUTHOR : DAP Application Group, L. Philipose, Dallas *
* CREATED 2004(C) BY TEXAS INSTRUMENTS INCORPORATED. *
* VERSION: 1.0 *
******************************************************************************

/* Include Header File */
#include "configcfg.h"
#include <csl_legacy.h>
#include "dc_conf.h"
/*Function Prototypes*/
void swiEnablePrephFunc();
extern unsigned short temp; /* Raw buffer for AD data */
int pingpong=0; /*=1 pingbuffers full */

/**************************************************************************************/
/* init_dsk() */
/* This initializes the EMIF */
/**************************************************************************************/
void init_dsk(void)
{
UINT32 gblctl,ce0ctl,celctl,ce2ctl,ce3ctl,sdctl,sdtim,sdext;
/* initialization of the EMIF */

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

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

celctl = EMIF_MK_CECTL( 0, 2, 0, 0, 0, 0, 0, 0);

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

/* PERIOD, XFR */
sdtim = EMIF_MK_SDTIM( 1562, 0);
sdext = EMIF_SDEXT_NA;

/* make CE2 control register value */
/* This is the CE space used by the ADS8371. */
/* Use the timing values from dc_conf.h: */

ce2ctl = EMIF_MK_CECTL{
    EMIF_CECTL_RDHLDOF (RDHLDOF), /* read hold */
    EMIF_CECTL_MTYPE_ASYNC32,
    EMIF_CECTL_RDSRB_OF (RDSRB), /* read strobe */
    EMIF_CECTL_TA_NA,
    EMIF_CECTL_RDSETUP_OF (RDSETUP), /* read setup */
    EMIF_CECTL_WRHLDOF (WRHLDOF), /* write hold */
    EMIF_CECTL_WRSTROB_OF (WRSTROB), /* 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() */

/**************************************************************************/
/** init_adc() */
/** This initializes the ADC */
/**************************************************************************/

void init_adc()
{
   int i;

   /* Initialize A/D */
   temp = *ADS8371_CONVSTZ; /*Discard the first three conversions*/
   for (i=0; i<=22; i++){}; /*after power up.*/
   temp = *ADS8371_RD;
   for (i=0; i<=1; i++){};
   temp = *ADS8371_CONVSTZ;
   for (i=0; i<=21; i++){};
   temp = *ADS8371_RD;
   for (i=0; i<=1; i++){};
   temp = *ADS8371_CONVSTZ;
   for (i=0; i<=21; i++){};
   temp = *ADS8371_RD;
}

/**************************************************************************/
/** hwiDMA_isr(): */
/** Hardware Interrupt Function disables EDMA channels */
/** and Timer1, Then post software interrupt. */
/**************************************************************************/

void hwiDMA_isr()
{
   if (pingpong==1){
      TIMER_pause(hTimer1);
      EDMA_disableChannel(hEdmaCh2); /*Disable EDMA channel 2-CONVST#*/
      EDMA_disableChannel(hEdmaCh7); /*Disable EDMA channel 7 -RD#*/

      //Uncomment next line to go again
      //swiEnablePrephFunc();
   }
   pingpong=!pingpong;
   IRQ_reset(IRQ_EVT_EDMAINT); /*Reset EDMA interrupt */
   IRQ_enable(IRQ_EVT_EDMAINT);
Appendix B

```c
EDMA_intDisable(TCCINTNUM6); /*Disable EDMA interrupt */
EDMA_intClear(TCCINTNUM6); /*Clear EDMA interrupt */
EDMA_intEnable(TCCINTNUM6); /*Enable EDMA interrupt */
}

/******************************
/*swiEnablePrephFunc:
*/
/* Software Interrupt Function configures EDMAC2 and */
/* EDMAC7, enables respective EDMA channels and Timer1 */
/*******************************/
void swiEnablePrephFunc()
{
    IRQ_enable(IRQ_EVT_EDMAINT);
    EDMA_intDisable(TCCINTNUM6); /*Disable EDMA interrupt */
    EDMA_intClear(TCCINTNUM6); /*Clear EDMA interrupt */
    EDMA_intEnable(TCCINTNUM6); /*Enable EDMA interrupt */
    EDMA_config(hEdmaCha7,&edmaCfgChan7);
    EDMA_config(hEdmaCha2,&edmaCfgChan2);
    EDMA_enableChannel(hEdmaCha2); /*Enable EDMA channel 2 -CONVST# */
    EDMA_enableChannel(hEdmaCha7); /*Enable EDMA channel 7 -RD# */
    EDMA_clearChannel(hEdmaCha7);
    TIMER_start(hTimer1); /*Start A/D CONVST# trigger */
}

/*******************************/
/* End Functions.c */
/*******************************/
```
IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI’s standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

<table>
<thead>
<tr>
<th>Products</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amplifiers</td>
<td>Audio</td>
</tr>
<tr>
<td>Data Converters</td>
<td>Automotive</td>
</tr>
<tr>
<td>DSP</td>
<td>Broadband</td>
</tr>
<tr>
<td>Interface</td>
<td>Digital Control</td>
</tr>
<tr>
<td>Logic</td>
<td>Military</td>
</tr>
<tr>
<td>Power Mgmt</td>
<td>Optical Networking</td>
</tr>
<tr>
<td>Microcontrollers</td>
<td>Security</td>
</tr>
<tr>
<td></td>
<td>Telephony</td>
</tr>
<tr>
<td></td>
<td>Video &amp; Imaging</td>
</tr>
<tr>
<td></td>
<td>Wireless</td>
</tr>
</tbody>
</table>

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

Copyright © 2005, Texas Instruments Incorporated