

# Interfacing the ADS8383 to the TMS320C6711 DSP

Lijoy Philipose

Data Acquisition Applications

#### ABSTRACT

This application note presents a software and hardware interface of the ADS8383 18-bit 500 kHz analog-to-digital converter to the TMS320C6711 DSP. The hardware platform used to develop this application is ADS8383EVM and TMS320C6711 DSK. The software developed reads 1024 blocks of samples continuously from ADS8383. In an effort to reduce development time, the source code is available on TI's website. Project collateral discussed in this application note can be downloaded from the following URL: http://www.ti.com/lit/zip/SLAA174.

1	Introduction				
2	Hard	ardware	3		
	2.1	1 TMS320C6711 DSK	3		
	2.2	2 ADS8383EVM	3		
	2.3	B Hardware Interface	3		
3	Soft	oftware Interface	4		
	3.1	EDMA Setup	4		
	3.2	2 Timer Setup			
	3.3	B HWI Setup	24		
	3.4	SWI Setup			
	3.5	5 LOG Setup			
Арр	endix	dix A MAIN.C			
Арр	endix	dix B Functions.C	34		

#### **List of Figures**

1 Hardware Connection	. 4
2 EDMA Chan2 General Tab	5
3 EDMA Chan2 Operation Mode Tab Options	6
4 EDMA Chan2 Source Tab Options	7
5 EDMA Chan 2 Destination Tab Options	7
6 EDMA Chan2 Transfer Complete Tab Options	8
7 EDMA Chan2 Transfer Count Tab Options	8
8 EDMA Chan2 Transfer Index Tab Options	9
9 EDMA Chan2 Link Tab	10
10 EDMA Chan2 Advanced Tab Options	. 11
11 EDMA Chan6 General Tab Options	. 12

Trademarks are the property of their respective owners.

12	EDMA Chan6 General Tab Options	. 13
13	EDMA Chan6 Operation Mode Tab Options	. 13
14	EDMA Chan6 Source Tab Options	. 14
15	EDMA Chan6 Destination Tab Options	. 15
16	EDMA Chan6 Transfer Complete Tab Options	. 16
17	EDMA Chan6 Transfer Count Tab Options	. 16
18	EDMA Chan6 Index Tab Options	. 17
19	EDMA Chan6 Link Tab Options	. 18
20	EDMA Chan6 Advanced Tab Options	. 19
21	EDMA Chan6 EXTINT6 Properties Tab	. 20
22	Timer0 Cfg Clock Control Tab Options	. 21
23	Timer0 Cfg Pin Control Tab Options	21
24	Timer0 Cfg Counter Control Tab Options	22
25	Timer0 Cfg Advanced Tab Options	. 23
26	TimerCfg0 Mapped in Timer0 Device Tab Options	. 24
27	HWI_INT8 General Tab Options	. 25
28	HWI_INT8 Dispatcher Tab Options	. 26
29	swiEnablePreph Properties	. 27
30	swiFormat18bit Properties	. 28
31	LOG Screen	. 29
32	Trace Properties Tab	. 30
33	Functions and Flow	. 31



### 3 Introduction

The ADS8383 is a single channel, 500KSPS, 18-bit analog-to-digital converter.<sup>M</sup>The converter can easily interface to the TMS320C6711 digital signal processor (C6711 DSP). For system development the TMS320C6711 DSP starter kit (DSK) and ADS8383EVM was used. The program written uses the EDMA channels and timer to trigger, capture and store blocks of data from the converter. A soft copy of the software for this application note is available for download from TI's website.

## 4 Hardware

The hardware platform employs the TMS320C6711 DSK and ADS8383EVM and some discrete logic.

### 4.1 TMS320C6711 DSK

The TMS320C6711 DSP starter kit (DSK) not only provides an introduction to C6000 technology, but is powerful enough to use for fast development of networking, communications, imaging and other applications like data acquisition. See TI's website (<u>www.ti.com</u>) for more information on the C6711 DSK.

### 4.2 ADS8383EVM

The ADS8383 evaluation module is an easy way to test both the functional and dynamic performance of this 18-bit analog-to-digital converter. The evaluation module includes only those circuits essential to showing the performance of the converter, reference, power, and input circuits. The digital inputs and outputs are buffered. The analog signal can be applied via standard 0.1inch IDC header/socket or via SMA connector. The buffered data bus is available via 0.1inch IDC header/socket. The ADS8383 control input is also made available standard 0.1-inch IDC header/socket. Therefore this EVM can be plugged into any development system for rapid prototyping. For information on this product, search www.ti.com for the ADS8383EVM.

### 4.3 Hardware Interface

One ADS8383 was mapped into memory space CE2, so the CE2 signal was used as CS for the A/D device. The read and convert start signals were generated by using a 2–4 decoder mapped into CE2. Performing a read operation to CE2 address space generates the RD and CONVST pulses. In this case a reading from address 0xA0028000 generates a read pulse and reading from 0xA002C000 generates a conversion start pulse. The C6711 has a 32-bit data bus; therefore BYTE was tied low allowing for 18-bit bus operation. The BUSY signal is first inverted and then applied to the external interrupt pin 6. The inversion of the BUSY is necessary because the EDMA interrupt controller recognizes only rising edges of external interrupts. The inversion is not necessary if the CPU is used to service A/D interrupts, because it can be programmed to trigger on a rising or falling edge. Finally, the data bus is mapped LSB-to-LSB. The hardware connections are shown in Figure 1.

C6711 DSP and C6000 DSP are trademarks of Texas Instruments.



Figure 1. Hardware Connection

# 5 Software Interface

The program utilizes two EDMA channels to read from 0xA0028000 and 0xA002C000 to generate the conversion start and read signals to the converter. The first EDMA channel is synchronized by Timer0, generating a periodic conversion start signal. The second EDMA channel is triggered by an external interrupt 6 event, it generates the read signal. Once all 1024 samples have been captured and stored in the array *ad\_buffer*, the EDMA controller generates an interrupt to the DSP. The interrupt service routine disables all activity with the converter, processes the data and posts a software interrupt. The function associated with that software interrupt copies the data from *ad\_buffer*, clears the high 14 bits of the 32-bit word and places the data into an array called *Fbuffer*. The function then posts another software interrupt that enables the EDMA channels and timer0 to capture another 1024 samples. See Figure 33 for software flow and how the individual functions are called.

The software interface is written entirely in C using Code Composer Studio<sup>™</sup> (CCS) IDE using CSL and DSP/BIOS. This application report assumes readers have a basic understanding of DSP/BIOS and CSL as found in the help system of CCS. If not, it is recommended they go through the tutorials in Code Composer Studio before proceeding further. The program is written using the configuration tool.

The following sections describe how the configuration tool is used for setting the EDMA, timer, hardware interrupts, software interrupts, and LOG objects.

### 5.1 EDMA Setup

- 1. Expand the chip support library (CSL) module in the configuration tool. Right click and insert two edmaCfg objects. Rename those objects to edmaCfgChan2 and edmaCfgChan6 (see Figure 2).
- 2. EDMA channel 2 is used to trigger a new conversion. Right click on edmaCfgChan2. In the General tab, you may choose to enter a comment saying it is a start conversion channel. In the operation mode tab select the options shown in Figure 3. Enabling Frame Sync causes an EDMA transfer when a Timer0 event occurs. In this case it is a 16-bit read into a variable called temp.



👯 Config				
Estimated Data Size: 2973 Est. Min. Stack Size (M/	edmaCfgChan2 Pr	operties		×
⊕- 🙀 System	Transfer Complet	e 📔 Transfer Count	Index Link	. Advanced
🕀 📻 Instrumentation	General	Operation Mode	Source	Destination
🗄 📲 Scheduling			· ·	l l
	comment: start	conversion		
Input/Output	,			
CSL - Chip Support Library				
EDMA Enhanced Direct Memory Access				
EDMA Configuration Manager				
edmaCfgChan2				
🕀 📅 EDMA Resource Manager				
🔄 📅 Parameter RAM Table Entry				
EMIF - External Memory Interface				
EMIFA - External Memory Interface A (6				
EMIFB - External Memory Interface B (64				
MCBSP - Multichannel Buffered Serial Por				
Witerbi Decoder Coprocessor - VCP		OK (	Cancel <u>Apply</u>	Help
WRUS - Expansion Rus				

Figure 2. EDMA Chan2 General Tab

edmaCfgChan2 Pr	operties		X
Transfer Comple General	te Transfer Count Operation Mode	Index Link	Advanced Destination
Frame Sync(FS)	Enable	•	
Element Size(ESI	ZE) 16-bit	•	
Priority Levels(PR	II) Low	•	
	ΟΚ	Cancel Applu	Help

### Figure 3. EDMA Chan2 Operation Mode Tab Options

- 3. In the Source tab, select the options as shown in Figure 4. The 16-bit reads are from address 0xA002C000. Toggling these particular address lines causes the decoder to generate a CONVST pulse.
- 4. In Destination tab, select the options shown in Figure 5. The 16-bit value read into by the EDMA channel is stored in the temp variable. This object is externally declared. The value on the data bus is discharged, because it is a not read from the converter.



edmaCfgChan2 Prope	ties					×
Transfer Complete General	Transfer Count Operation Mode		dex   Source	Link	Advanced Destination	
Two Dimensional Sou	ce Transfer(2DS)		Disable	<b>-</b>		
Source Address Upda	te Mode(SUM)		None	•		
Peripheral Device Tra	nsfer Srce(PDTS)		Disable	7		
Source Address Forma	at 🛛	Numeric		•		
Src Addr - Enter Nur	neric value :	0xA00	2000			
Src Addr - Extern De	cl. Symbol name :	NULL				
Src Addr - Enter full	address :	NULL				
Src Addr - Enter Har	idle Name :	NULL				
	ОК	Cancel		Apply	Help	

Figure 4. EDMA Chan2 Source Tab Options

edmaCfgChan2 Properties								
Transfer Comple General	te Transfer Count Operation Mode	Index     Source	Link e	Advanced Destination	1			
Two Dimensional	Two Dimensional Destination Transfer(2DD		) Disable					
Destination Addre	ess Update Mode(DUM)	No	ne	•				
Peripheral Device	e Transfer Dest.(PDTD)	Dis	sable	<b>V</b>				
Destination Addre	ess Format	Extern Dec	І. ОЫј.	•				
Dst Addr - Ente	r Numeric value :	0x000000	)00					
Dist Addri - Exte	rn Decl. Symbol Name :	temp						
Dst Addr - Ente	r full address :	NULL						
Dst Addr - Ente	r Handle Name :	NULL						
	ОК	Cancel	Apply	Help				

Figure 5. EDMA Chan 2 Destination Tab Options

- 5. In the Transfer Complete tab, choose options shown in Figure 6. Transfer complete interrupt is disabled.
- 6. In the Transfer Count tab, choose options shown in Figure 7. Frame count is 0x0100.

dmaCfgChan2 Prop	erties			×
General Transfer Complete	Operation Mode Transfer Count	Source	∍   Link	Destination Advanced
Transfer Complete I	nterrupt(TCINT)		D	isable 🔽
Transfer Complete (	Code(TCC):		2	
Most Significant Bit	s of Transfer Complete Co	ode(TCCM):	0	
Alternate Transfer C	iomplete Interrupt(ATCIN	T)	D	isable 🔽
Alternate Transfer C	iomplete Code(ATCC):		0	
				1
	OK Ca	incel	Apply	Help

Figure 6. EDMA Chan2 Transfer Complete Tab Options

edmaCfgChan2 Pr	operties			×
General Transfer Comple	Operation Mode te Transfer Count	Source	 Link	Destination   Advanced
Transfer Counter	Transfer Counter Format			
Frame Cnt(FC)	Enter 16-bit value:	0x0100		
Frame Cnt(FC)	Enter Num or Symbol valu	e ; NULL		
Elt Counter(EC)	- Enter 16-bit value:	0x0001		
Elt Count(EC) -	Enter Num or Sym value :	NULL		
Element Count R	eload(ECRLD):	0x0100		
	OK Ca	ancel	Apply	Help

### Figure 7. EDMA Chan2 Transfer Count Tab Options

7. In the Index and Link tabs, choose options shown in Figure 8 and Figure 9.



H <mark>a</mark> Conlig.cdb	edmaCfgChan2 Properties		×
Estimated Data Size: 2973 Est. Min. SI Scheduling CLK - Clock Manager PRD - Periodic Function Ma HWI - Hardware Interrupt Se SWI - Software Interrupt Ma SWI - Software Interrupt Ma SUL - Idle Function Manager JDL - Idle Function Manager Synchronization Input/Output CSL - Chip Support Library CSL Extern Declaration DMA Direct Memory Access EDMA Enhanced Direct Me	edmaCfgChan2 Properties General Operation M Transfer Complete Transfer Index Format Frame Index(FIX)- 16-bit: Frame Index(FIX)- Enter Flum or Element Index(EIX) - 16-bit: Element Index(EIX) - Enter Flum	Mode Source Destination r Count Index Link Advance Numeric 0x0000 r Symbol value NULL 0x0000 n or Sym value NULL	x i i i i i i i i i i i i i
EDMA Configuration Ma edmaCfgChan2 edmaCfgChan6 EDMA Resource Manag Parameter RAM Table E EMIF - External Memory Inte EMIFA - External Memory Inte EMIFB - External Memory Inte	OK	Cancel Apply Hel	Þ

Figure 8. EDMA Chan2 Transfer Index Tab Options

U Contig odb	dmaCfgChan2 Prop	erties		×
Estimated Data Size: 2973 Est. Min. SI	General	Operation Mode	Source	Destination
🖻 🕞 Scheduling	Transfer Complete	Transfer Count	Index Li	nk Advanced
🕀 😲 CLK - Clock Manager	Service Service		and the state of the state	
PRD - Periodic Function Ma	Linking Event(LINK)	Disable		
H     H	and the second second			
TSK - Task Manager	Link Format	Li able Numbe	st 🔽	
IDL - Idle Function Manager	Link Handle Table	PARAMTEL NOT	HIT	
⊕ 🐼 Synchronization				
🗄 📠 Input/Output	Link to Table Numbe	r  9-	and the second	
🗄 🐺 CSL - Chip Support Library				
💼 🚞 CSL Extern Declaration 📲				
🕀 🏧 DMA Direct Memory Access				
EDMA Enhanced Direct Me				
EDMA Configuration Ma				
admaCtaChan2				
Parameter BAM Table F				
EMIF - External Memory Inte				
🗄 🚮 EMIFA - External Memory In				
🗄 🚮 EMIFB - External Memory In				
		OK Ca	ancel Ap	ply Help

Figure 9. EDMA Chan2 Link Tab

- 8. The Advanced tab should look like Figure 10. CONVST pulse is mapped in address 0xA002C000. The 16-bit data is stored in an externally declared object *temp*.
- 9. To map this configuration object to the resource, select and expand the EDMA resource manager. Right click on EDMA\_channel1\_TINTO and select properties. In the general tab, select those options shown in Figure 11. Create an EDMA channel handle called *hEdmaTint0* and enable pre-initialization of EDMA channel 1 registers. In the main program, the user needs to enable the channel. The initialize is done before entering into the main() function.



TEXAS

INSTRUMENTS

Figure 10. EDMA Chan2 Advanced Tab Options

Contig cdb					
Estimated Data Size: 2973 Est. Min. Stack Size (MAU	Js): 880	EDMA_C	hannel1_TINT(	) properties	
🖨 📅 EDMA Resource Manager	-	Property		Value	NAMES AND DESCRIPTION OF
EDMA_Channel0_DSPINT		comment	t	Timer 0 Interr	upt
- 🚼 EDMA_Channel1_TINT0		Open ED	MA Channel	True	
EDMA_Channel2_TINT1	EDMA C	hannel1	TINTO Prope	erties	×
EDMA_Channel3_SDINT	witter Mittania a R	Contraction of the	Sector Contraction of		
EDMA_Channel4_EXTINT4_GP	General				And Anna and Anna
EDMA_Channel5_EXTINT5_GP		Self-self-	(Constantine Constant)	Constant Street of Street	in the second
EDMA_Channel6_EXTINT6_GP	comme	ent	Timer 0 Inter	rupt	and the second second
THE FOMA_Channel/_EXTINT/_GP		EDUA	-		
EDMA_Channel8_TCC8_GPINT		en EDMA I	Lnannei		
EDMA_channels_fccs_dFinit	Handle		hEdmaTint0	1000	
EDMA_channello_rccto_dFir				Construction of the second sec	in the second
EDMA_channel12_XEVT0	🔽 En	able Pre-In	itialization		
EDMA_channel13_BEVT0					Stand Streets in
EDMA Channel14 XEVT1	Pre-ini	tialize with:	leamatrgtha	anz 🔟	
EDMA Channel15 REVT1	En En	able Select	ted Channel		
EDMA Channel16	Sector				
EDMA_Channel17_XEVT2					
EDMA_Channel18_REVT2	0	K	Cancel	Apply	Help
EDMA_Channel19_TINT2		and a start of the start of the start			
	•				

Figure 11. EDMA Chan6 General Tab Options

- 10. Right click on edmaCfgChan6 object and select properties. In the General tab, the user may choose to write that this object is used to read data from the converter.
- 11. In the Operation Mode tab, select options shown in Figure 13. Enable Frame sync, this triggers a read operation whenever an interrupt occurs signaling the end of a conversion. In this application, the BUSY pulse is inverted before being applied to the External interrupt 6 pin. This inversion is necessary before the EDMA recognizes only rising edges and falling edges. Remember that BUSY goes high when conversion begins, and returns low when conversion is complete.



### Figure 12. EDMA Chan6 General Tab

edmaCfgChan6 P	roperties		×
Transfer Comple General	ete Transfer Count Operation Mode	Index Link	Advanced Destination
Frame Sync(FS)	Enable		
Element Size(ES	IZE) 32-bit	•	
Priority Levels(Pf	RI) High	•	
	ОК	Cancel Apply	Help

Figure 13. EDMA Chan6 Operation Mode Tab Options

- 12. In the Source tab, set the options shown in Figure 14. A read pulse is generated by the decoder when accessing 0xA0028000. So the source address is 0xA0028000. Disable two-dimensional source transfer and source address update.
- 13. In the Destination tab, select the options shown in Figure 15. Disable the two dimensional Source transfer and select increment destination address. The data read from the converter is stored in an externally declared array called *ad\_buffer*. This is a 32-bit read and the converter is mapped DSP LSB to A/D LSB, therefore the top 14-bits of the 32-bit word should be masked out.

Transfer Complete	Transfer Count	Index	Link	Advanced
General	Operation Mode	Source		Destination
Two Dimensional S	ource Transfer(2DS)	Disable	◄	
Source Address Up	date Mode(SUM)	None	•	
Peripheral Device T	ransfer Srce(PDTS)	Disable	7	
Source Address For	mat N	umeric	•	
Src Addr - Enter N	umeric value :	0xA0028000		
Src Addr - Extern	Decl. Symbol name :	NULL		
Src Addr - Enter fi	ill address :	NULL		
Src Addr - Enter H	andle Name :	NULL		
	οκ ο	ancel	Apply	Help

Figure 14. EDMA Chan6 Source Tab Options

edmaCfgChan6 Prope	rties				×	
Transfer Complete General	Transfer Count Operation Mode	Index Source	Link. e	Advanced Destination		
Two Dimensional Destination Transfer(2DD)						
Destination Address U	pdate Mode(DUM)	Inc	rement	•		
Peripheral Device Tra	nsfer Dest.(PDTD)	Dis	sable	<b>T</b>		
Destination Address F	ormat	Extern Dec	I. ОЫј.	•		
Dst Addr - Enter Nur	neric value :	0x000000	)00			
Dist Addri - Extern De	ecl. Symbol Name :	ad_buffer				
Dst Addr - Enter full	address :	NULL				
Dst Addr - Enter Har	ndle Name :	NULL				
	ОК Са	ancel	Apply	Help		

Figure 15. EDMA Chan6 Destination Tab Options

- 14. In the Transfer Complete tab select the options as shown in Figure 16. Enable Transfer Complete Interrupt to interrupt the DSP. The controller interrupts the DSP, and transfers complete code 6 when all 0x0400 samples have been read and stored.
- 15. In the Transfer Count tab, select options shown in Figure 17. Frame Count is set to 0x0400, indicating a total transfer of 1024 samples.

General	Operation Mode	Source	•	Destination		
Transfer Complete	Transfer Count	Index	Link	Advanced		
Transfer Complete Interrupt(TCINT)						
Transfer Complete	Code(TCC):		6			
Most Significant B	ts of Transfer Complete C	ode(TCCM):	0			
Alternate Transfer	Complete Interrupt(ATCIN	(T)	Di	isable 🔽		
Alternate Transfer	Complete Code(ATCC);		0			

Figure 16. EDMA Chan6 Transfer Complete Tab Options

edmaCfgChan6 Pr	operties		X
General Transfer Complet	Operation Mode e Transfer Count	Source Source	Destination Advanced
Transfer Counter	Format	Numeric	
Frame Cnt(FC) -	Enter 16-bit value:	0x0400	
Frame Cnt(FC) -	Enter Num or Symbol valu	e : NULL	
Elt Counter(EC)	- Enter 16-bit value:	0x0001	
Elt Count(EC) -	Enter Num or Sym value :	NULL	
Element Count R	eload(ECRLD):	0x0001	
	OK Ca	ancel Apply	Help

### Figure 17. EDMA Chan6 Transfer Count Tab Options

16. In the Index and Link Tabs, select options shown in Figure 18 and Figure 19.



👯 Conlig odb	edmaCfgChan6 Prop	erties		×
Estimated Data Size: 2973 Est. Min. Stack Size	General	Operation Mode	Source	Destination
📄 🧝 Scheduling	Transfer Complete	Transfer Count	Index Link	Advanced
🗄 💮 CLK - Clock Manager				
📲 🥙 PRD - Periodic Function Manager	Index Format		Numeric	
🕀 🕛 HWI - Hardware Interrupt Service Rc				
🕀 🐺 SWI - Software Interrupt Manager	Frame Index(FIX)-16	S-bit:	0x0000	S. Alignetter
🔢 🕀 👷 TSK - Task Manager	The second second second	-	RITE I	Contraction of the second s
DL Idle Function Manager	Frame indexicity)	Enter Num of Symbol V	aide : Troch	are the second
	Element Index(EIX)	16-bit:	0x0000	and the second se
Enter Input/Uutput				nine (manager and states)
Emilian USL - Chip Support Library	Element Index(El>	) - Enter Num or Sym va	alue NULL	
DMA Direct Memory Access				
EDMA Enhanced Direct Memory Access				
edmaCfgChan6				and the second second
EDMA Resource Manager				
Parameter RAM Table Entry				
🗄 🕋 EMIF - External Memory Interface				
📋 🚋 🚮 EMIFA - External Memory Interface A				
👔 🗄 🕋 EMIFB - External Memory Interface B				
		UK Ca	ancel Apply	Help

Figure 18. EDMA Chan6 Index Tab Options

Hig Config odb	edmaCfgChan6 Prop	erties		al and the second	×
Estimated Data Size: 2973 Est. Min. Stack Size	General	Operation Mode	Source		Destination
🖻 🧱 Scheduling	Transfer Complete	Transfer Count	Index	Link	Advanced
🕀 😲 CLK - Clock Manager	Sector sector sector	and the second second			
PRD - Periodic Function Manager	Linking Event(LINK)	Disable	<b>•</b>		
🕀 🗒 HWI - Hardware Interrupt Service Ro	- And the second se				
	Link Format	Table Numb	et 🝸		
🕀 👷 TSK - Task Manager					
DL - Idle Function Manager	Link Handle Lable:	PARAMISE_NU	HU		
🕀 🙀 Synchronization	Link to Table Number	ne literation	and the second second		
⊡ 📠 Input/Output	Entry of dole freme				
□ CSL - Chip Support Library					
					Const Les Levies
DMA Direct Memory Access					
EDMA Enhanced Direct Memory Acc					
EDMA Configuration Manager					
edmaCfgChan2					Sector Sector
redmaCfgChan6					
EDMA Resource Manager					
Parameter RAM Table Entry					Contraction of the
EMIF - External Memory Interface					
EMIFA - External Memory Interface A					
EMIFB - External Memory Interface B				Annia	U-la I
		UNU	ance	EPPPy	

Figure 19. EDMA Chan6 Link Tab Options

- 17. In the Advanced tab, set the options shown in Figure 20. Read 1024 samples from the A/D converter for which the Read signal is mapped to 0xA0028000. When all 1024 samples have been stored, interrupt the DSP.
- 18. The configuration object now needs to be mapped into the appropriate resource. In the EDMA Resource Manager, right click on EDMA\_Channel6\_EXTINT6\_GPINT6 and select properties. In the General tab, select the options shown in Figure 21. Create hEdmaCha6 handle and preinitialize EDMA channel 6 registers with edmaCfgChan6. Channel initialization completes before entering the main() function. In the main function the channel needs to be enabled.



🗮 Contig edb	edmaCfgChan6 Properties	X
Estimated Data Size: 2973 Est. Min. Stack Size Scheduling CLK - Clock Manager PRD - Periodic Function Manager HWI - Hardware Interrupt Service Rc SWI - Software Interrupt Manager TSK - Task Manager TSK - Task Manager IDL - Idle Function Manager Synchronization CSL - Chip Support Library CSL - Chip	GeneralOperation ModeTransfer CompleteTransfer CountOption:Source Address FormatSource Address - Numeric :Transfer Counter FormatTransfer Counter - Numeric:Destination Address FormatDestination Address - NumericIndex FormatIndex register - Numeric:Element Count Reload and Link Address	Source Destination   Index Link   Advanced     0x20350001   Numeric   0x04000001   Extern Decl. Obj.   0x00000000   Numeric   0x00000000   0x00000000   0x00000000
EMIFB - External Memory Interface B	ОК	Cancel Apply Help

Figure 20. EDMA Chan6 Advanced Tab Options

Estimated Data Size: 2973 Est. Min. Stack Size (MAUs): 880       EDMA_Channel6_EXTINT6_GPINT6_GPINT6 properties         Fig. EDMA_Channel0_DSPINT       Fig. EDMA_Channel0_DSPINT         Fig. EDMA_Channel0_DSPINT       Fig. EDMA_Channel1_TINT0         Fig. EDMA_Channel3_SDINT       EDMA_Channel3_SDINT         Fig. EDMA_Channel5_EXTINT4_GPINT4       Fig. EDMA_Channel5_EXTINT5_GPINT5         Fig. EDMA_Channel5_EXTINT5_GPINT5       EDMA_Channel6_EXTINT6_GPIN6_GPIN6_GPIN6_GPIN6_GPIN6_GPIN6_GPIN6_GPIN6	🕂 Config cdb	
EDMA Resource Manager       Property       Value         EDMA_Channel0_DSPINT       EDMA_Channel0_DSPINT       External Interrupt Pin 6/G         EDMA_Channel1_TINT0       EDMA_Channel2_TINT1       EDMA_Channel3_SDINT         EDMA_Channel4_EXTINT4_GPINT4       EDMA_Channel5_EXTINT5_GPINT5       EDMA_Channel5_EXTINT5_GPINT5         EDMA_Channel5_EXTINT5_GPINT5       EDMA_Channel7_EXTINT7_GPINT7       EDMA_Channel7_EXTINT7_GPINT7         EDMA_Channel7_EXTINT7_GPINT7       EDMA_Channel1_TCC10_GPINT2       General         EDMA_Channel1_TCC1_GPINT3       Comment       External Interrupt Pin 6/G         EDMA_Channel7_EXTINT5_GPINT5       General       Comment         EDMA_Channel1_TCC1_GPINT3       General       Comment         EDMA_Channel1_TCC1_GPINT3       EDMA_Channel13_REVT0       EDMA_Channel13_REVT1         EDMA_Channel16       EDMA_Channel17_XEV72       EDMA_Channel18_REV72         EDMA_Channel18_REV72       EDMA_Channel18_REV72       Enable Selected Channel	Estimated Data Size: 2973 Est. Min. Stack Size (MAUs): 880	EDMA_Channel6_EXTINT6_GPINT6 properties
EDMA_Channel15_REVT1     International 20 whith fedmacige hand       EDMA_Channel16     EDMA_Channel17_XEVT2       EDMA_Channel18_REVT2     EDMA_Channel19_TINT2       OK     Cancel	Estimated Data Size: 2973 Est. Min. Stack Size (MAUs): 880 EDMA Resource Manager EDMA_Channel0_DSPINT EDMA_Channel1_TINT0 EDMA_Channel3_SDINT EDMA_Channel4_EXTINT4_GPINT4 EDMA_Channel5_EXTINT5_GPINT5 EDMA_Channel6_EXTINT5_GPINT5 EDMA_Channel8_TCC8_GPINT0 EDMA_Channel9_TCC9_GPINT1 EDMA_Channel10_TCC10_GPINT2 EDMA_Channel12_XEVT0 EDMA_Channel13_REVT0 EDMA_Channel14_XEVT1 EDMA_Channel14_XEVT1	EDMA_Channel6_EXTINT6_GPINT6 properties         Property       Value         comment       External Interrupt Pin 6/G         Open EDMA Channel       True         Handle       hEdmaCha6         Enable Pre-Initialization       True         MA_Channel6_EXTINT6_GPINT6 Properties       X         eneral
	EDMA_Channel15_REVT1	Pre-Initialize with:     edmaCrgChanb       Enable Selected Channel       OK     Cancel

Figure 21. EDMA Chan6 EXTINT6 Properties Tab

### 5.2 Timer Setup

The Timer setup can also be done via the configuration tool. Expand the CSL-Chip Support Library section, and expand Timer–Timer Device module. Right click on Timer Configuration Manager and select Insert timerCfg. Rename timerCfg to timerCfg0 to map it into Timer0.

- 1. Right click on timerCfg0 and go to the Clock Control tab. Select options shown in Figure 22. Input Clock source is CPU clock divided by 4. Select Clock mode.
- 2. In the Pin Control tab, select options as shown in Figure 23. Function of TOUT is Timer Output and TOUT is uninverted.



Config	
Estimated Data Size: 2973 Est. Min. Stack Size (MAUs): 880	timerCfg0 properties
🗄 🔐 Instrumentation	Property Value
🗄 🕵 Scheduling	comment Trigger CONVSTz
🕀 🕵 Synchronization	Input Clock Source (CLKSR (CPU clock)/4
	Pulse Width (PWID) Two Clock cycles
CSL - Chip Support Library	Function of TOUT (FUNC) Timer Output
CSL Extern Declaration	timerCfg0 Properties
	Pin Control   Counter Control   Advanced
EDMA Enhanced Direct Memory Access	General Clock Control
🖻 📅 EDMA Configuration Manager	
- 📲 edmaCfgChan2	Input Clock Source (CLKSRC)
	Clock/Pulse Mode (CP) Clock mode
EDMA Resource Manager	
Tarameter RAM Table Entry	Pulse Width (PWID) Two Clock cycles 🗾
H. MIF - External Memory Interface	
EMIER - External Memory Interface B (64x devices o	
EMILE - External Memory Internate B (04x devices of     Emilia Memory Internate B (04x devices of     Emilia Memory Internate B (04x devices of	
Turbo Decoder Coprocessor -TCP	
TIMER - Timer Device	
🛱 🎬 TIMER Configuration Manager	
timerCfg0	OK Cancel Apply Help
🗄 🔛 🏧 TIMED Decource Manager	

Figure 22. Timer0 Cfg. Clock Control Tab Options

ti	imerCfg0 Propertie	5			×	
	General Pin Control	Cour	Clo nter Control	ck Control   Advanced		
	Function of TOUT (	FUNC)	Timer Out	put	J	
	DATA output (TC	UT pin):	0			
	TOUT Inverter Control (INVOUT)					
	TINP Inverter Co	ntrol (INVII	NP)	Uninverted	3	
	ОК	Cancel	Apply	Help		

Figure 23. Timer0 Cfg. Pin Control Tab Options

3. In the Counter Control tab, type the value shown in Figure 24. Type 0x26 into the Period Value field. The frequency of the timer is set by the formula below:

Frequency =  $\frac{[f(clock source)]}{(2 \times timer period register)}$ 

In this application note, the C6711 DSP is clocked at 150MHz. In the previous tab, clock divide-by-4 option was chosen as the source for timer0. Therefore the sampling frequency is set for

Frequency =  $\frac{(150e6/4)}{38}$  = 493 kHz.

The Advanced tab should read as shown in Figure 25.

timerCfg0 Propertie	:5	×		
General	Cloc	k Control		
Pin Control	Counter Control	Advanced		
Timer Operation	Hold			
Period Value :	0x00000026			
Counter value (optional): 0x00000000				
ОК	Cancel Apply	Help		

Figure 24. Timer0 Cfg. Counter Control Tab Options

ti	merCfg0 Properties		X
	General	Clock Control	
	Pin Control	Counter Control Advanced	
	Control Register (CTL) :	0x00000311	
	Period Register (PRD) :	0x00000026	
	Counter Register (CNT)	: 0x0000000	
	OK Ca	ancel <u>Apply</u> Help	

#### Figure 25. Timer0 Cfg. Advanced Tab Options

4. Now that the configuration object has been set, it must be mapped into the appropriate device. Expand the Timer Resource Manager and right click on Timer\_Device0. Select options as shown in Figure 26. Enable timer0 registers to be preinitialized and open a hTimer0 handle.

Config	_ <b>_ _ _</b>
Estimated Data Size: 2973 Est. Min. Stack Size (MAUs): 880	Timer_Device0 properties
CSL - Chip Support Library  CSL Extern Declaration  CSL Extern Declaration  externad_buffer  externtemp  DMA Direct Memory Access	Property Value     comment Timer 0     Open Timer Device True     Handle hTimer0     Enable Pre-Initialization True     Pre-Initialize with timerCfg0
EDMA Ennanced Direct Memory Access EDMA Configuration Manager - 닭양 edmaCfgChan2 - 닭양 edmaCfgChan6	Timer_Device0 Properties
EDMA Resource Manager     Barameter RAM Table Entry     Barameter RAM Table Entry	comment: Timer 0
Turbo Decoder Coprocessor -TCP      TIMER - Timer Device      TIMER - Timer Device	Handle : hTimer0  Fendble Pre-Initialization  Pre-Initialize with: timer0fa0
TIMER Configuration Manager	
Timer_Device1	OK Cancel Apply Help

Figure 26. TimerCfg0 Mapped in Timer0 Device Tab Options

#### 5.3 HWI Setup

The interrupt routines can be mapped into the appropriate location in the interrupt vector table using the configuration tool.

In this application note only the EDMA controller interrupt is enabled. Under Scheduling, expand the Hardware Interrupt tab (HWI). Right click on HWI\_INT8 and select properties. In the General tab, select the options shown in Figure 27. Interrupt source is EDMA\_controller. Interrupt Routine is hwiDMA\_isr function. In the Dispatcher tab, select options shown in Figure 28. Use Dispatcher, and enable self interrupt mask.





Figure 27. HWI\_INT8 General Tab Options

🖽 Config edb				
Estimated Data Size: 2973 Est. Min	. Stack Size (MAUs): 880	HWI_INT8 propertie	s	
🖻 📴 Scheduling	<b>_</b>	Property	Value	
① ① ① ① ① ① ① ① ① ① ① ① ① ① ① ① ①	UV/I_INTO Properties	L comment	defines the	INT8 Interrupt
HWI - Hardware Interru	General Dispatcher			sr h
HWI_NMI	Use Dispatcher			addr)
	Arg:	0x0000000		b
	Interrupt Mask	self		
🕛 HWI_INT7	Interrupt Bit Mask:	0x0100		
<sup>e</sup> , Hwi_int8 <sup>e</sup> , Hwi_int9	Don't modify cache	control		le
·····································	Program Cache Control	Mask cacha	enable 💌	
·····································	Data Cache Control Mar	k cache	enable 💌	
HWI_INT14	OK Can	cel <u>A</u> pply	Help	
E SWI - Software Interrup ▲				

Figure 28. HWI\_INT8 Dispatcher Tab Options

### 5.4 SWI Setup

Software Interrupts are mapped under the Schedule tab in the configuration tool. Right click on SWI-software interrupt manager and select insert SWI twice.

- 1. This creates two SWI objects. Rename the objects to swiEnablePreph and swiFormat18 bit.
- 2. Right click on swiEnablePreph and set the fields shown in Figure 29. EnablePreph software interrupt enables the timers, and re-initializes and enables EDMA channels.
- 3. Right click on swiFormat18bit and set fields as shown in Figure 30. Format18bit software interrupt copies the data from ad\_buffer to F\_Buffer. While copying data into F\_Buffer, the data is formatted such that the top 14 bits of the 32-bit word are cleared.



W <mark>.</mark> Config.edb		Contraction of the		
Estimated Data Size: 2973 Est. Min. Stack Size (MAUs): 880		swiEnablePreph properties		
<ul> <li>System</li> <li>Instrumentation</li> <li>Scheduling</li> <li>CLK - Clock Manager</li> <li>PRD - Periodic Function Manager</li> <li>HWI - Hardware Interrupt Servis</li> <li>SWI - Software Interrupt Manager</li> <li>SWIEnablePreph</li> <li>swiEnablePreph</li> <li>swiFormat18bit</li> <li>SYNChronization</li> <li>Input/Output</li> <li>CSL - Chip Support Library</li> </ul>	wiEnablePreph General Comment: function: priority mailbox: O arg0: Oxt	Property comment function priority mailbox Properties dd comments here> wiEnablePrephFunc 1	Value <add comments="" here=""> _swiEnablePrephFunc 1 0 0 0 0 0 0 0 0 0 0 0 0 0</add>	
	arg1: Ox(	Cancel	Apply Help	

Figure 29. swiEnablePreph Properties



Figure 30. swiFormat18bit Properties

### 5.5 LOG Setup

The LOG object is a under the Instrumentation tab of the configuration tool. It behaves as a *printf* function in C. Unlike *printf*, this function updates the user screen when the CPU is in an idle state. It is useful in debugging software.

 Right click on LOG-Event Log Message and select insert LOG as shown in Figure 31. In the general tab, set the options shown in Figure 32. Create an internal circular buffer of 512 words long.



🗮 Config edb			
Estimated Data Size: 2973 Est. Min. Stack Size (MAUs): 880	trace properties		
System Instrumentation LOG - Event Log Manager LOG_system STS - Statistics Object Manager Scheduling Synchronization Input/Output CSL - Chip Support Library	Property comment bufseg buflen (words) logtype datatype format	Value <add comments<br="">INTERNAL 512 circular printf 0x%x, 0x%x, 0x%x</add>	

Figure 31. LOG Screen

W Config.cdb				_OX
Estimated Data Size: 2973 Est. Min. Stack Size (MAUs): 880		trace properties		
<ul> <li></li></ul>		Property comment bufseg	Value	
E - C IIII STS - E - C IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	General comment:	Kadd comments here>		circular printf Ox%x, Ox%x, Ox%x
⊡@ Input/Oul ⊕⊊ CSL - Chi	bufseg: buflen (words) logtype datatype format	INTERNAL		
	ОК	Cancel Ap	Jy Help	

Figure 32. Trace Properties Tab Options



Figure 33. Functions and Flow

# Appendix A MAIN.C

```
/* File: main.c
                                                            */
                                                            */
/*
    Description: Program for interfacing ADS8383
/*
     to an C6711 DSK. Program use the Timer0 to trigger a
                                                            */
/* convst# pulse, at about 493kHz using EDMA channel 2.
                                                            */
/* Inverted BUSY signal is used to trigger EDMA channel 6 to
                                                           */
/\star 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 timer0 and post software
                                                            */
/* interrupt with will mask off the top 14 bits of 32 bit word.
                                                            */
/* The data is stored in Fbuffer. The program then repeats.
                                                            */
/*
                                                            */
/* AD converter address: CE2 memory space
                                                            */
/*
                                                            */
                0xA0028000 (Read#)
/*
                 0xA002C000 (Convst#)
                                                            */
/* Hardware Connections:
                                                            */
/* CS# => CE2#
                                                            */
/* RD# => Generated from 2-4 decoder mapped to 0xA0028000
                                                            */
/* CONVST# => Generated from 2-4 decoder mapped at 0xA002C000
                                                           */
/* BUSY => Inverted then wired to EXTERNAL INT6
                                                            */
/* AUTHOR : DAP Application Group, L. Philipose, Dallas
                                                            */
/*
    CREATED 2003(C) 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(ad buffer,BLOCK SZ);
#pragma DATA ALIGN(Fbuffer,BLOCK SZ);
unsigned int ad_buffer[BLOCK_SZ]; /* raw data from AD, written to by EDMA */
unsigned int Fbuffer[BLOCK SZ]; /* masked buffer of AD */
unsigned int temp, n=0;
void main(void)
{
  int i;
  /* initialize the EMIF*/
  init dsk();
  IRQ reset(IRQ EVT EDMAINT);
                                     /*Reset EDMA interrupt
                                                            */
  EDMA intDisable(TCCINTNUM);
                                      /*Disable EDMA interrupt */
  EDMA intClear (TCCINTNUM);
                                      /*Clear EDMA interrupt
                                                            */
                                     /*Enable EDMA interrupt */
  EDMA intEnable(TCCINTNUM);
                                     /*Initialize data buffers */
  for (i=0; i<BLOCK SZ; i++) {
     Fbuffer[i]=0xFFFFF;
       ad buffer[i]=0xFFFFF;
   }
/*Configuration of Timer0, EDMA channels 2 and 6 where during before entering
 main.c */
/*Only need to enable them in main. */
  TIMER start(hTimer0);
                                     /*Start A/D CONVST# trigger */
  EDMA_enableChannel(hEdmaTint0);
                                     /*Enable EDMA channel 2 -CONVST# */
  EDMA enableChannel(hEdmaCha6); /*Enable EMDA channel 6 -RD# */
  /* Enable the EDMA controller interrupt */
  IRQ enable(IRQ EVT EDMAINT);
   LOG printf(&trace, "GO! \n"); /*Go sample at 490 kHz*/
}
/* end main.c
                                                      * /
```

# Appendix B Functions.C

```
/* File: functions.c
                                                      */
/* Description: Functions for interfacing ADS8383 to C6711
                                                      */
/* init dsk, swiFormat18bitFunc, hwiDMA isr, swiEnablePrephFunc */
/* AD converter address: CE2 memory space
                                                      */
/*
                                                      */
               0xA0028000 (Read#)
/*
                                                      */
               0xA002C000 (Convst#)
/* Hardware Connections:
                                                      */
/* CS# => CE2#
                                                      */
/* RD# => Generated from 2-4 decoder mapped to 0 \times A0028000
                                                      */
/* CONVST# => Generated from 2-4 decoder mapped at 0xA002C000
                                                      */
/* BUSY => Inverted then wired to EXTERNAL INT6
                                                      */
/* AUTHOR : DAP Application Group, L. Philipose, Dallas
                                                     */
/*
   CREATED 2003(C) BY TEXAS INSTRUMENTS INCORPORATED.
                                                      */
/* VERSION: 1.0
                                                      */
/* Include Header File */
#include "Configcfg.h"
#include "dc conf.h"
#include <csl legacy.h>
extern unsigned int ad buffer[BLOCK SZ]; /* Raw buffer for AD data
                                                              */
extern unsigned int Fbuffer[BLOCK SZ], cnt=0; /* Buffer for masked AD data */
/* 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);
   /* RDHLD, MTYPE, RDSTRB, TA, RDSETUP, WRHLD, WRSTRB, WRSETUP */
   ceOctl = EMIF MK CECTL( 0, 3, 0, 0, 0, 0, 0, 0);
```

```
/* RDHLD, MTYPE, RDSTRB, TA, RDSETUP, WRHLD, WRSTRB, WRSETUP */
   celctl = 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 ADS8383 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 */
                                     /* write hold */
       EMIF_CECTL_WRHLD OF (WRHLD),
       EMIF CECTL WRSTRB OF (WRSTRB),
                                     /* write strobe */
       EMIF CECTL WRSETUP OF (WRSETUP) /* write setup */
   );
   /* configure the EMIF */
   EMIF ConfigB(gblctl, ceOctl, ce1ctl, ce2ctl,
       ce3ctl,sdctl,sdtim,sdext);
   return;
} /* end init dsk() */
/*swiFormat18bitFunc:
                                                      */
/*
     Software Interrupt Function resets the upper 14 bits */
/* of the 32-bit word and stores 18-bit A/D samples
                                                      */
/* in Fbuffer
                                           */
void swiFormat18bitFunc()
{
```

TEXAS INSTRUMENTS

```
int i=0;
    for (i=0; i<BLOCK SZ; i++) {</pre>
      Fbuffer[i] = 0x0003FFFF & ad buffer[i];
 }
  SWI post(&swiEnablePreph);
}
*/
/*hwiDMA isr():
/* Hardware Interrupt Function disables EDMA channels and */
                                          */
/* Timer0, Then post software interrupt.
void hwiDMA isr()
{ cnt++;
11
   LOG printf(&trace, "hwiDMA isr \n");
  /* Clear the pending interrupt from the EDMA interrupt pending register */
  EDMA intClear(TCCINTNUM);
  TIMER pause (hTimer0);
  EDMA disableChannel(hEdmaCha6);
  EDMA disableChannel(hEdmaTint0);
 swiFormat18bitFunc();
}
*/
/*swiEnablePrephFunc:
/*
    Software Interrupt Function configures EDMAC2 and
                                          */
/*
    EDMAC6, enables respective EDMA channels and Timer0
                                          */
void swiEnablePrephFunc()
{
 EDMA config(hEdmaTint0, &edmaCfgChan2);
  EDMA config(hEdmaCha6, &edmaCfgChan6);
 TIMER start(hTimer0);
 EDMA enableChannel(hEdmaTint0);
  EDMA enableChannel(hEdmaCha6);
}
/* 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:

Products		Applications	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments

Post Office Box 655303 Dallas, Texas 75265

Copyright © 2003, Texas Instruments Incorporated

#### 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 Copyright © 2019, Texas Instruments Incorporated