

# ***Interfacing the TMS320C5402 DSP to the TLV2541 ADC and the TLV5636 DAC***

*Lijoy Philipose*
*AAP Data Conversion*

## **ABSTRACT**

This report shows how to interface the TLV2541 and TLV5636 12-bit serial-data converters to the TMS320C5402™ DSP via the McBSP. It presents a hardware solution and a software solution to interface the 16-bit fixed-point TMS320C5402 DSP to read from the TLV2541 12-bit, 200-KSPS, single-channel, serial analog-to-digital converter (ADC), and to write to the TLV5636 12-bit digital-to-analog converter (DAC).

## **Contents**

|          |                                                         |           |
|----------|---------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b> .....                               | <b>2</b>  |
| <b>2</b> | <b>The Hardware</b> .....                               | <b>2</b>  |
| 2.1      | TMS320C5402 DSK Starter Kit .....                       | 2         |
| 2.2      | The Multiconverter EVM .....                            | 3         |
| <b>3</b> | <b>The Software</b> .....                               | <b>4</b>  |
| 3.1      | DSP .....                                               | 4         |
| 3.2      | CPLD .....                                              | 4         |
| 3.2.1    | DSP CNTL2 Control Register (I/O Address = 0x0004) ..... | 4         |
| 3.2.2    | Write to CPLD Register CNTL2 .....                      | 4         |
| 3.3      | McBSP .....                                             | 5         |
| 3.4      | TLV2541 .....                                           | 7         |
| 3.5      | TLV5636 .....                                           | 8         |
| 3.6      | Hardware Overview .....                                 | 9         |
| 3.7      | Software Overview .....                                 | 9         |
| <b>4</b> | <b>References</b> .....                                 | <b>11</b> |
|          | <b>Appendix A TLV2541.ASM</b> .....                     | <b>12</b> |
|          | <b>Appendix B C5402 Registers</b> .....                 | <b>17</b> |
|          | <b>Appendix C C5402 Vector Table</b> .....              | <b>20</b> |
|          | <b>Appendix D C5402 Memory Mapping</b> .....            | <b>24</b> |

## **List of Figures**

|   |                                                       |    |
|---|-------------------------------------------------------|----|
| 1 | McBSP Register-Addressing Scheme .....                | 5  |
| 2 | TLV5636 Serial Data Format .....                      | 8  |
| 3 | TLV5636 Control Register .....                        | 8  |
| 4 | Hardware-Interface Block Diagram .....                | 9  |
| 5 | Software Interface Flowchart .....                    | 10 |
| 6 | TLV2541 and TLV5636 Interfaced to the C5402 DSP ..... | 11 |

## List of Tables

|   |                                                                          |   |
|---|--------------------------------------------------------------------------|---|
| 1 | Analog-to-Digital Serial-Data Converters on the Multiconverter EVM ..... | 3 |
| 2 | Digital-to-Analog Serial-Data Converters on the Multiconverter EVM ..... | 3 |
| 3 | CNTL2 Control Register Bit Definitions .....                             | 4 |
| 4 | McBSP Register Settings .....                                            | 7 |
| 5 | TLV5636 Register-Select Bits .....                                       | 8 |
| 6 | TLV5636 Program-Data Bits .....                                          | 8 |
| 7 | Reference Bits .....                                                     | 8 |

## 1 Introduction

The TLV2541 is a low-power, 200 KSPS, 12-bit serial analog-to-digital converter. Interfacing the TLV2541 and TLV5636 to the C5402 DSP is a simple task. The ADC does not require any input configuration. The device can be plugged into a DSP system and expected to perform right away.

The TLV5636 is a 12-bit serial digital-to-analog data converter. Although this DAC requires some configuration, it is also easily interfaced to the C5402 DSP. This application report describes how to interface the ADC and the DAC to the same McBSP port on the C5402 DSP. The assembly code developed for this interface is provided in the appendixes.

## 2 The Hardware

The C5402 DSK starter kit and the Multiconverter EVM were used to develop this report.

### 2.1 TMS320C5402 DSK Starter Kit

The C5402 DSK is specially designed for digital communications applications and comes complete with a TMS320C5402-based target board, DSK-specific *Code Composer Studio* debug tools, 32K application-size-limited C-compiler/assembler/linker, parallel-port interface, power supply, and cables.

The C5402 DSP device features 100-MHz clock, 40-bit ALU, 16K x 16-bit dual-access on-chip RAM, 4K x 16-bit on-chip ROM, and advanced multibus architecture with three separate 16-bit data-memory busses and one program-memory bus. In addition to these features, the DSK has an embedded JTAG emulation via the TBC and the IEEE-1284 parallel ports. The onboard parallel-port controller allows the host PC to use this port for emulation, or to directly access the host-port interface of the C5402 DSP. Additional features include onboard standard JTAG-interface connection for optional emulation, and expansion connectors for add-on accessories. Texas Instruments now provides expansion connectors or adapter boards for all data-converter EVMs to interface to this DSK.

Enhanced peripherals of particular interest to this report are the McBSP serial ports. The McBSP is explained further in Chapter 3.

The C5402 DSK supports a TMS320C5402 DSP which can operate at frequencies up to 100 MHz with a core voltage of 1.8 V and an I/O voltage of 3.3 V. The DSK provides support for all the DSP interfaces and control signals. The JTAG-emulation interface is used to support both embedded and external JTAG emulations. The control interface is used to reset the device and to provide external interrupts. The McBSP0, by default, is used to interface to a telephone DAA circuit. This port is also available to the daughterboard via an onboard multiplexer. Another default use for the McBSP1 is in microphone/speaker interfaces. It is also brought to the peripheral-expansion connector for daughterboard use. The CPLD controls the source of McBSP0 and McBSP1.

## 2.2 The Multiconverter EVM

The Multiconverter EVM was developed by Texas Instruments to help customers evaluate families of 8-pin ADCs and DACs. Tables 1 and 2 describe all the data converters that can be tested with this EVM. Refer to the Multiconverter EVM user's guide for more information on hardware configurations.

**Table 1. Analog-to-Digital Serial-Data Converters on the Multiconverter EVM**

| PART NUMBER | NUMBER OF CHANNELS | INPUT               | MAXIMUM THROUGHPUT |
|-------------|--------------------|---------------------|--------------------|
| TLV2541     | 1                  | Unipolar            | 200 KSPS           |
| TLC2551     | 1                  | Unipolar            | 400 KSPS           |
| TLV2542     | 2                  | Unipolar            | 200 KSPS           |
| TLC2552     | 2                  | Unipolar            | 400 KSPS           |
| TLV2545     | 1                  | Pseudo differential | 200 KSPS           |
| TLC2555     | 1                  | Pseudo differential | 400 KSPS           |

**Table 2. Digital-to-Analog Serial-Data Converters on the Multiconverter EVM**

| PART NUMBER | RESOLUTION (BITS) | NUMBER OF DACs | INTERNAL REFERENCE |
|-------------|-------------------|----------------|--------------------|
| TLV5624     | 8                 | 1              | Yes                |
| TLV5623     | 8                 | 1              | No                 |
| TLV5606     | 10                | 1              | No                 |
| TLV5616     | 12                | 1              | No                 |
| TLV5636     | 12                | 1              | Yes                |
| TLV5617A    | 10                | 2              | No                 |
| TLV5618A    | 12                | 2              | No                 |
| TLV5625     | 8                 | 2              | No                 |
| TLV5626     | 8                 | 2              | Yes                |
| TLV5637     | 10                | 2              | Yes                |
| TLV5638     | 12                | 2              | Yes                |

## 3 The Software

The DSP, McBSP, and CPLD devices must be initialized before attempting to read or write to the data converters.

### 3.1 DSP

The 'VC5402 DSK provides the DSP with a single 20-MHz frequency reference via the DSP built-in crystal oscillator. The DSP's clock-mode pins are configured via DIP-switch settings to allow for a number of different frequencies, up to the parts maximum rate of 100 MHz. The CLKMD register can also be changed after reset to adjust the DSP's operating frequency. The CPU-clock frequency is 100 MHz when CLKMD is equal to 0x4007.

### 3.2 CPLD

There are seven DSP CPLD registers mapped to the DSP's lower I/O address space starting at address 0x0000. Only control register 2 (CNTL2) is of interest to this report.

#### 3.2.1 **DSP CNTL2 Control Register (I/O Address = 0x0004)**

This register selects the source of data for both McBSPs. Bit 1 of this register needs to be modified; otherwise McBSP0 defaults to the onboard device as its input source. For this report McBSP0 needs to be set to use the daughterboard as its source of data. Table 2 shows the register-bit definitions.

**Table 3. CNTL2 Control Register Bit Definitions**

| BIT | NAME      | R/W | DESCRIPTION                                                    |
|-----|-----------|-----|----------------------------------------------------------------|
| 7   | DAAOH     | RW  | DAA off-hook control (0 = on-hook, 1 = off-hook)               |
| 6   | DAACID    | RW  | DAA caller ID enable (0 = disabled, 1 = enabled)               |
| 5   | FLASHENB- | RW  | Select FLASH =1 (default) or SRAM (=0) for external memory (1) |
| 4   | INT1SEL   | RW  | Interrupt 1 source selection (0 = UART, 1 = daughterboard)     |
| 3   | FC1CON    | RW  | MIC/speaker AD50 FC control bit (0)                            |
| 2   | FC0CON    | RW  | DAA AD50 FC control bit (0)                                    |
| 1   | BSPSEL1   | RW  | McBSP1 select control (0 = mic/speaker, 1 = daughterboard)     |
| 0   | BSPSEL0   | RW  | McBSP0 select control (0 = TelSet DAA, 1 = daughterboard)      |

#### 3.2.2 **Write to CPLD Register CNTL2**

Control register 2 needs to be set to enable the daughterboard as the source for McBSP0. This is accomplished by setting CNTL2=0x0001. This register is mapped in I/O space 4h; the port instruction should be used to access this register:

```
port(#4h) = #0001h ; /*Write 0x0001 to CNTL2 register */
```

### 3.3 McBSP

The multichannel buffered serial port (McBSP) is a superset of the standard serial ports found on Texas Instruments digital-signal processors (DSP). In addition to features found on the previous serial-port interfaces, the McBSP is able to directly interface to TI/E1 framers, IOM–2 compliant devices, MVIP switching-compatible and ST–BUS-compliant devices, AC97-compliant devices, IIS-compliant devices, and SPI devices. It provides a wide selection of transmit/receive data sizes,  $\mu$ -Law and A-Law companding, programmable polarity for both frame synchronization and data clocks, and highly-programmable internal clock and frame generation. These features and programming requirements are described in the Texas Instruments document *TMS320C54x DSP Enhanced Peripherals Reference Set, Volume 5*.

First let us look how the McBSP registers are accessed. The McBSP registers are memory-mapped using a register subaddressing scheme. Figure 1 shows a visual representation of this scheme. Register subaddressing involves multiplexing a set of registers to a single location in the memory map. A sub-bank address register is used to control the multiplexer. A subdata register (SPSDx) is used to read or write data to the desired subaddressed register. To access a specific subaddressed register, the register’s subaddress location is written into the subaddress register (SPSAx). This directs the multiplexer to connect to the desired physical location in memory. When a write access occur, data written to the subdata register is moved to the embedded data register specified by the subaddress register. Similarly for a read access, the contents of the register specified in the subaddress register are moved to the subdata register.

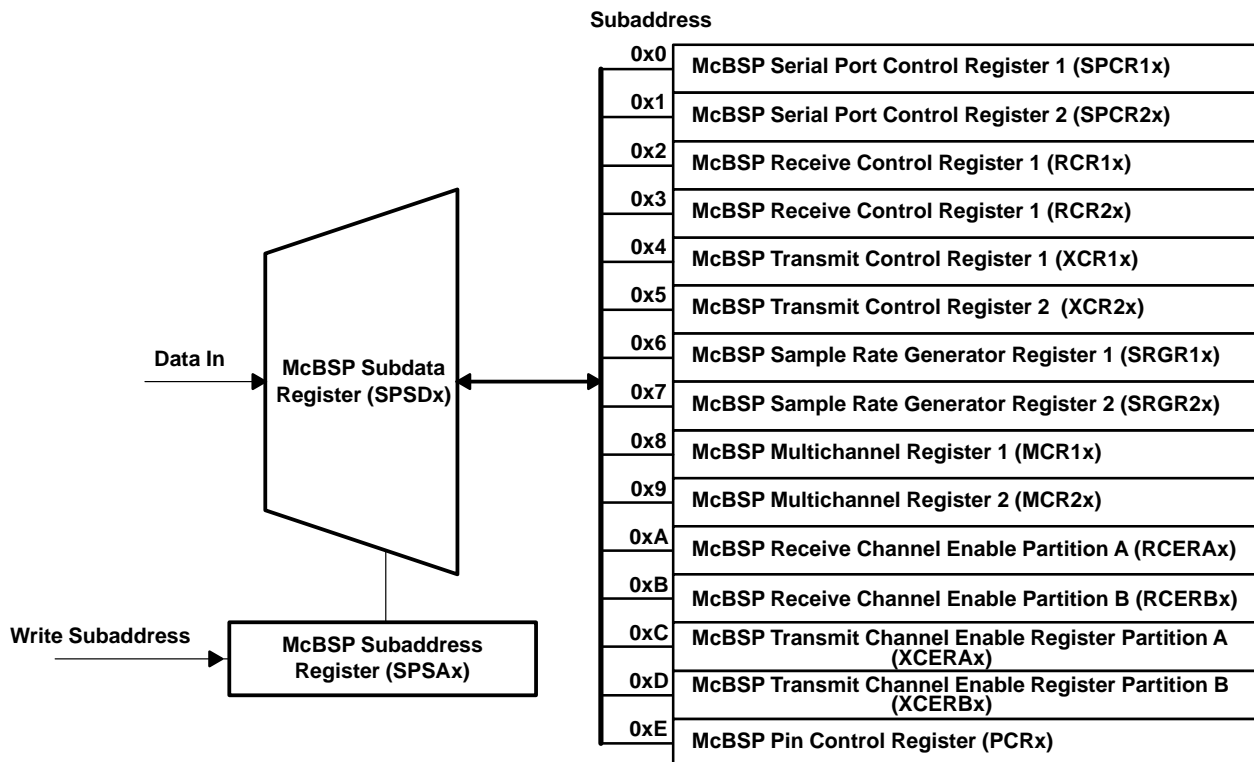


Figure 1. McBSP Register-Addressing Scheme

Let us use the McBSP0 as an example: the subdata register (SPSD) is at location 0x039 and the subaddress register (SPSAx) is at location 0x038 in physical memory. The following assembly-code sample writes 0x000 to the serial-port control register 1 of McBSP0.

```
SPSA0      .set      038h    ;McBSP0 subaddress register
SPSD0      .set      039h    ;McBSP0 subdata register
SPCR10_SUB .set      000h    ;McBSP0 serial-port control register 1 subaddress
mmr(#SPSA0) = #SPCR10_SUB
mmr(#SPSD0) = #000h
```

There are 16 registers associated with each McBSP. Interfacing a single TLV2541 ADC or TLV5636 DAC to the McBSP requires the proper configuration of only nine of these registers.

- The *serial port control register 1 (SPCR1)* contains the McBSP receiver status bits and the main switch to enable or disable the receiver. This register includes the clock-stop mode bit, which sets the serial port for various clocking modes for SPI and non-SPI schemes. Also included in SPCR1 is the ABIS-mode bit and the receiver-interrupt mode bit.
- The *serial port control register 2 (SPCR2)* contains the McBSP transmitter-status bits and the main switch to enable or disable the transmitter. This register also contains the bits to reset the frame-sync generator and the sample-rate generator.
- The *pin control register (PCR)* contains the bits to configure the McBSP pins as inputs or outputs during normal serial-port operation. This register is used to reconfigure the serial-port pins as general-purpose inputs or outputs when the receiver or transmitter is disabled. The PCR configures the transmitter and receiver clock and frame-sync modes. For example, these bits determine whether CLKX/R and FSX/R are input or output pins and what their polarity is.
- *Receive control register 1 (RCR1)* contains the bits to configure various receiver options. The value of this register determines the receiver-word size (between 8 and 32 bits) and the number of words per frame (1 to 128) expected per receiver event.
- *Receive control register 2 (RCR2)* determines the size of the word received and the bit delay after the frame-sync pulse. This register configuration plays an essential role when transfers greater than 16 bits and multiple phases are necessary. The register bit of interest to this application is the data-bit delay. The RCR2 register bits also select between  $\mu$ -law, A-law companding, and whether the MSB or the LSB is transferred first in noncompanding 8-bit transfers.
- The *transmit control register 1 (XCR1)* contains the bits that determine the transmit-word length and frame size. A transfer can be 8 to 32 bits wide and anywhere from one to 128 words long.
- *Transmit control register 2 (XCR2)* contain the bits that determine the transmit-data delay and select between  $\mu$ -law, A-law companding, and whether the MSB or the LSB is transferred first in noncompanding 8-bit transfers.

- *Sample-rate generator register one (SRGR1) and sample-rate generator register 2 (SRGR2)* control the sample-rate generator. The sample-rate generator is composed of a three-stage clock divider that allows programmable data clocks (CLKG) and framing signals (FSG). These are McBSP internal signals that can be programmed to drive the receive/transmit clock (CLKR/X) and the receive/transmit data framing (FSR/X). Sample-rate generator registers (SRGR[1,2]) control the operation of the various features of the sample-rate generator. These registers are used to control the width of the frame-sync pulse and to determine whether frame-sync is an external input driven by the sample-rate generator or a signal to indicate that data has been copied from DXR[1,2] to XSR[1,2]. These registers control whether the sample-rate generator clock is derived from the CPU clock or from the CLKX pin, and by what value to divide the CPU clock to produce the desired serial clock (CLKX/R).

All the McBSP register bits are describe in Texas Instruments document *TMS32054x DSP Enhanced Peripherals Reference Set, Volume 5*, literature number SPRU302. Table 4 summarizes the register settings used in this interface.

**Table 4. McBSP Register Settings**

| McBSP0 ADDRESS | McBSP0 SUBADDRESS | ACRONYM | REGISTER INITIALIZED | COMMENT                                                                                                                                 |
|----------------|-------------------|---------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 0021           |                   | DRR10   |                      | Receive data register                                                                                                                   |
| 0023           |                   | DXR10   |                      | Transmit data register                                                                                                                  |
| 0038           |                   | SPSA0   |                      | McBSP0 subaddressing register                                                                                                           |
| 0039           | 0x0000            | SPCR10  | 0x0001               | The LSB bit must be 0 (0x0000) to disable the transmitter while configuring the McBSP.                                                  |
|                | 0x0001            | SPCR20  | 0x02C1               | The LSB bit must be 0 (0x02C0) to disable the receiver while configuring the McBSP.                                                     |
|                | 0x0002            | RCR10   | 0x0040               | Selects one 16-bit word transfer per frame                                                                                              |
|                | 0x0003            | RCR20   | 0x0001               | Set 1-bit delay on receiver. Receiver assumes MSB arrives first on the next clock cycle after FSR pulse.                                |
|                | 0x0004            | XCR10   | 0x0040               | Selects one 16-bit word transfer per frame                                                                                              |
|                | 0x0005            | XCR20   | 0x0001               | Transmitter shifts out data immediately following falling edge of FSX.                                                                  |
|                | 0x0006            | SRGR10  | 0x0009               | The CLKX has a frequency of 10 MHz assuming 100-MHz CPU clock. CLKX=CPU clock / (CLKGDV +1) when CLKGDV=SRGR1[7,0]                      |
|                | 0x0007            | SRGR20  | 0x2000               | The sample-rate generator clock is derived from the CPU clock.                                                                          |
|                | 0x000E            | PCR0    | 0x0A00               | FSX is determined from sample-rate-generator frame-synchronization-mode bit SRGR2.FSGM. CLKX output is driven by sample-rate generator. |

### 3.4 TLV2541

The TLV2541 is a 12-bit, 200-KSPS analog-to-digital converter with a very-low power consumption. It provides a three-wire interface to a host of microprocessors, and a four-wire interface to DSPs. The packages for this device include a small eight-pin MSOP and an SOIC. The TLV2541 ADC is designed using successive-approximation and charge-redistribution DAC architecture. During a sixteen-bit serial transfer, the first twelve bits of the transfer contain the data, and the remaining four bits are zeros.

### 3.5 TLV5636

The TLV5636 is a 12-bit voltage-input DAC with programmable internal reference. It provides an interface that is compatible with both SPI and TMS320 serial ports. This device is available in 8-pin SOIC and 8-pin MSOP packages. The TLV5636 is programmed with a 16-bit serial string containing four control bits and twelve data bits, as illustrated in Figure 2. See Tables 5 and 6 for control-bit configuration.

**Data Bits**

|     |     |     |     |     |     |    |    |    |    |    |    |    |    |    |    |
|-----|-----|-----|-----|-----|-----|----|----|----|----|----|----|----|----|----|----|
| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| R1  | SPD | PWR | R0  | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**Figure 2. TLV5636 Serial Data Format**

**Table 5. TLV5636 Register-Select Bits**

| R1 | R0 | REGISTER                       |
|----|----|--------------------------------|
| 0  | 0  | Write data to DAC              |
| 0  | 1  | Reserved                       |
| 1  | 0  | Reserved                       |
| 1  | 1  | Write data to control register |

**Table 6. TLV5636 Program-Data Bits**

|                         | 0                | 1          |
|-------------------------|------------------|------------|
| Speed control bit (SPD) | Slow mode        | Fast mode  |
| Power control bit (PWD) | Normal operation | Power down |

The control register for the DAC allows users to select between different internal and external reference voltages, as shown in Table 7. The reference-select bits are bits 1 and 0 of the serial word; these bits are shown in Figure 3.

**Control Register**

|     |     |     |     |     |     |    |    |    |    |    |    |    |    |      |      |
|-----|-----|-----|-----|-----|-----|----|----|----|----|----|----|----|----|------|------|
| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1   | D0   |
| X   | X   | X   | X   | X   | X   | X  | X  | X  | X  | X  | X  | X  | X  | REF1 | REF0 |

**Figure 3. TLV5636 Control Register**

**Table 7. Reference Bits**

| REF1 | REF0 | REFERENCE |
|------|------|-----------|
| 0    | 0    | External  |
| 0    | 1    | 1.024 V   |
| 1    | 0    | 2.048 V   |
| 1    | 1    | External  |

### 3.6 Hardware Overview

The hardware interface for these two devices is shown in Figure 4. The general-purpose input/output pin (XF) is used to select both data converters. The inverter between the XF pin and the DAC ensures that only one device is selected at a time. Serial port lines CLKX/R, FSX/R, and DX/R are connected to both data converters. The attached source code assumes that the McBSP serial lines are tied to the ADC and the DAC.

Users can measure the throughput delay of the signal and observe signal integrity through the ADC–DSP–DAC system by probing the input analog signal and DAC output on an oscilloscope. The assembly code provided with the application note does the following: the user supplies a continuous analog input signal to the TLV2541 ADC; this signal gets converted and read out by the DSP. The DSP stores the sample and writes it directly to the TLV5636 DAC. This program continues infinitely, or until the user halts the DSP.

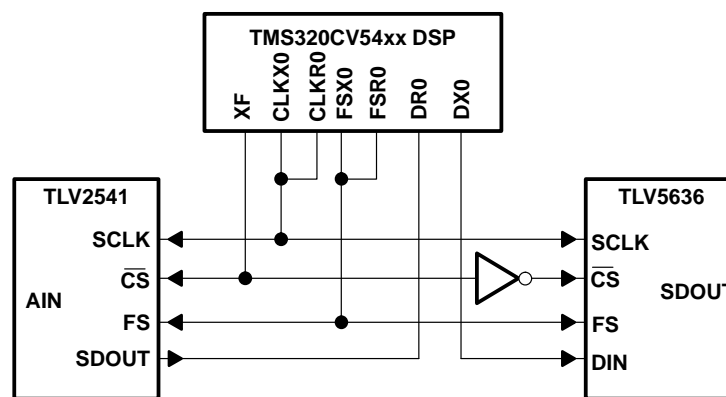


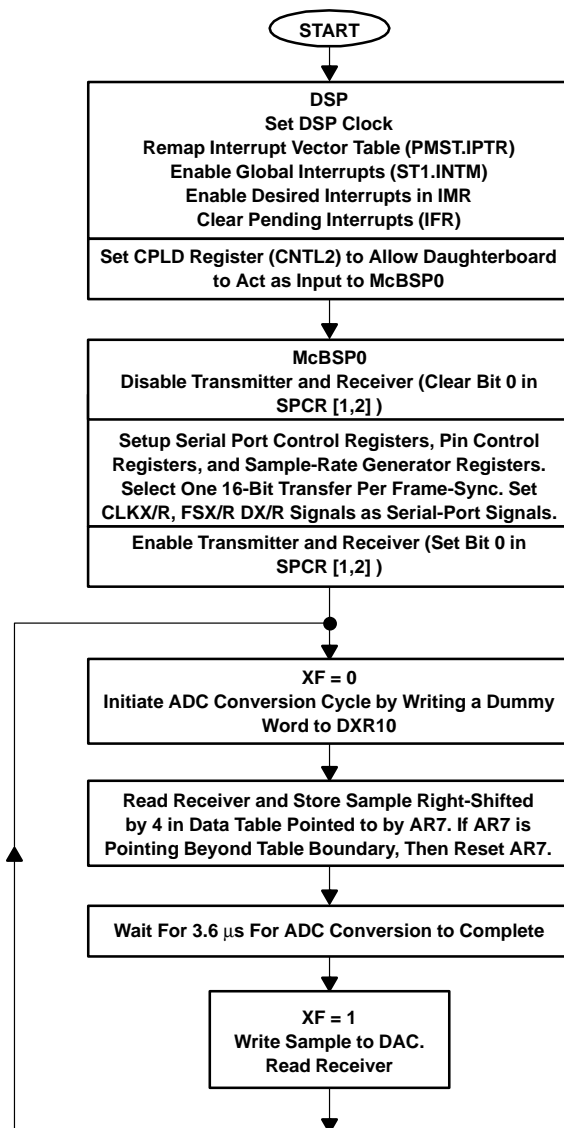
Figure 4. Hardware-Interface Block Diagram

### 3.7 Software Overview

The software interface flowchart for this devices is shown in Figure 4. Begin by setting the DSP registers

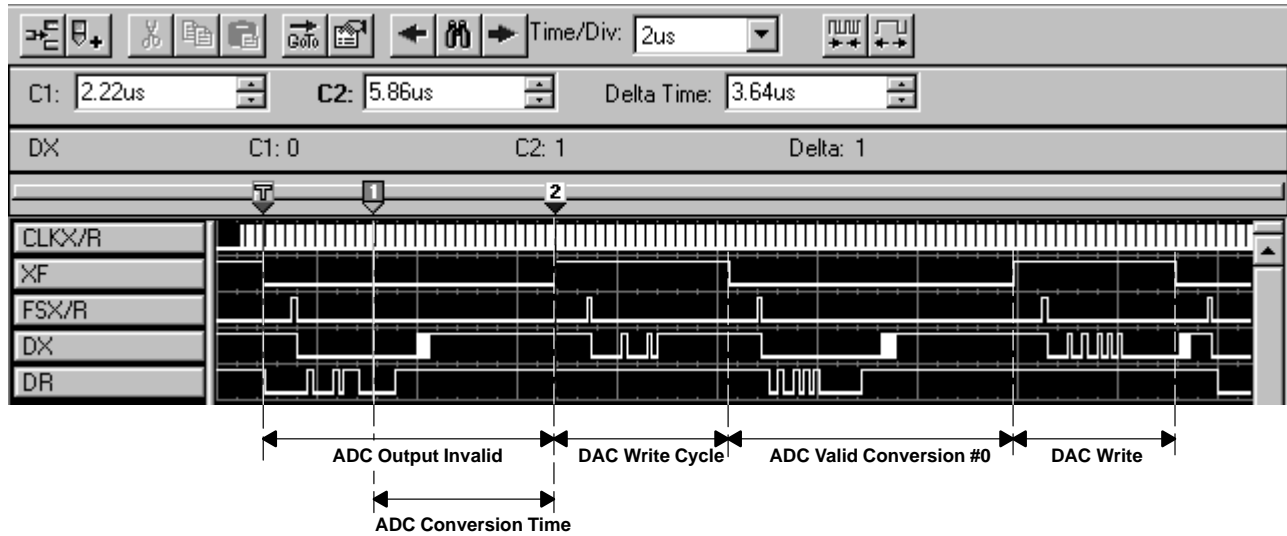
1. Set the C5402 DSP for 100-MHz operation.
2. Disable all interrupts.
3. Remap interrupt vector table to 0x0080 in program memory.
4. Disable all maskable interrupts.
5. Clear all pending interrupts.
6. Set wait-state register.

Next, the CPLD register must be set to allow the McBSP0 input source to come from the daughterboard, also known as the expansion bus. Then the McBSP0 registers need to be initialized as described in Table 4, McBSP Register Settings. The McBSP initialization procedures state that both receiver and transmitter must be disabled before configuring its respective register bits.



**Figure 5. Software Interface Flowchart**

The software interface for the single channel ADC is very simple. Users do not program the ADC—they just have to provide the proper control signals. In DSP mode, the data converter begins the conversion cycle and outputs data in response to the frame-sync pulse generated by a transmit operation. When the conversion is complete, users need to write a word to the transmit-data register of the McBSP. The serial port then generates a frame-sync pulse to synchronize the serial data it is shifting out of the transmit data pin. Since  $XF=0$ , the DAC does not see the data on the transmit data or on the frame-sync lines. The frame-sync will be received by the ADC and the converted data will be shifted out on the rising edges of the serial clock. The McBSP receiver is programmed to read the data on the falling edges of the serial clock. This setup ensures that the data is always valid when the receiver reads the line. Once the 16-bit transfer is complete, the ADC data lines will go to the high-impedance state. The ADC requires  $3.6\ \mu\text{s}$  to complete a conversion; this is accomplished by keeping the device selected for at least  $3.6\ \mu\text{s}$  after the 16-bit transfer is complete (see Figure 6). The conversion result will be invalid if the device is deselected ( $XF/\overline{CS}=1$ ) before the required time period.



**Figure 6. TLV2541 and TLV5636 Interfaced to the C5402 DSP**

The DAC is selected by setting XF=1. Once this is accomplished, a 16-bit DAC word can be written to the transmit-data register. During this transfer the DAC output pin has the analog equivalent of the digital sample written to it from the previous write cycle. The DAC can be deselected following the 16-bit transfer.

The program attached in the appendixes is written in algebraic assembly language that continuously reads the TLV2541ADC, stores the sample in the DSP, and then writes it out to the TLV5636 DAC.

## 4 References

1. *TMS320C54X DSP CPU and Peripherals Reference Set*, Volume 1, literature number SPRU1311
2. *TMS320C54X Optimizing C Compiler user's guide*, literature number SPRU103D
3. *TMS320C54XX DSP Enhanced Peripherals Reference Set*, Volume 5, literature number SPRU302
4. *TMS320C54XX DSP Reference Set*, Volume 3: Algebraic Instruction, literature number SPRU179B
5. *TMS320C54XX Assembly Language Tools user's guide*, literature number SPRU102D
6. *Multiconverter EVM user's guide*, literature number SLAU047
7. TLV2541 data sheet, literature number SLAS245
8. TLV5636 data sheet, literature number SLAS223

## Appendix A TLV2541.ASM

```
*****
* TITLE           : TLV2541 Interface routine                               *
* FILE            : tlv2541.ASM                                           *
* FUNCTION        : MAIN                                                  *
* PROTOTYPE       : void MAIN ()                                          *
* CALLS           : N/A                                                  *
* DESCRIPTION     : This program is to be used to interface the TLV2541 and TLV5636 *
*                  : to C5402 McBSP0. The routine continuously reads data in from ADC, *
*                  : stores the 12 bits at address beginning at 0x3200, then shifts *
*                  : that data back out to the DAC. This program loops infinitely. *
* AUTHOR          : AAP Application Group, L. Philipose, Dallas, Tx        *
*                  : CREATED 2000(C) BY TEXAS INSTRUMENTS INCORPORATED. *
* REFERENCE       : TMS320C54x User's Guide, TI 1997                      *
*                  : Data Acquisition Circuits, TI 1999                    *
*****
```

```

        .title    "TLV2541 ADC"
        .mmregs
        .width    80
        .length   55

        .sect ".vectors"
        .copy "C5402vec.asm"

        .sect ".data"
        .include regs.h

**Local and Global variables defined**
AD_DP      .usect ".varibl", 0 ;Data page label for variables.
ADWORD     .usect ".varibl", 1 ;Temporarily Store Sample in this variable
Temp       .usect ".varibl", 1 ;
        .sect ".text"
main:
***Initialize DSP***
        DP=#0;                ;Memory Mapped Registers in Data Page 0

        CLKMD =#0             ;Switch to DIV mode
TstStatu:
        A=CLKMD               ;Test Clock Mode Status
        A &= #01b             ;mask out PLL status bit
        if (ANEQ ) goto TstStatu ;wait until DIV mode is enabled

        CLKMD = #4007h        ;Set C5402 DSP clock to 100 MHz.

        INTM=#1               ;Disable all interrupts.
        SXM=#0                ;No Sign Extension
        CPL=0                 ;DP used for relative addressing
        PMST = #00F8h        ;Remap Interrupt Vector Table to 0x0080h in Program Memory

```

```

IMR = #0000h           ;Disable maskable interrupts
IFR = #0FFFFh         ;Clear all Pending Interrupts

SWWSR = #2000h        ;Need Wait-State of at least 2.

DP=#ADWORD            ;Change Data Page to where Variables are stored.
@Temp=#0003h
port(DSP_CPLD_CNTL2) =@Temp    ;Select daughterboard as source for McBSP0
XF = 1                 ;Set ADC Chip Select High
call McBSP0_init       ;Initialize McBSP0
INTM=#0               ;enable global interrupts

DP=#AD_DP             ;
AR7=#3200h            ;Address to store Samples at.

*Begin conversion cycle*

Begin_Cycle:
    XF = 0              ;Clear ADC_CS. ADC Chip Select is held low throughout
                        ;this program.
    A=#0000h           ;Loading Accum. A with 0x0 to generate FSX/R pulse.
    call McBSP0WriteRead
    B=B<<<-4           ;Shift data to right by 4
    @ADWORD=B          ;Move 12-bit word to ADWORD
    repeat(#280)       ;Do nothing. Allow enough time for conversion
    NOP

    AR0=#3400h         ;Table ends at 0x3400h
    DP=#ADWORD         ;Load Data Page Pointer with data page of ADWORD
    TC = (AR0 == AR7)  ;is AR7 = AR0? (table end reached?)
    if (TC) goto Reset_Data_Table ;If table end reached move reset pointer to
                        ;beginning of table
    *AR7+ = data(@ADWORD) ;Store 12-bit Sample at location pointed to by AR7
    goto SendtoDAC

Reset_Data_Table:
    AR7=#3200h         ;Point to first position in the data table
SendtoDAC:

    XF = 1             ;Select DAC
    A=@ADWORD
    call McBSP0WriteRead ;Write sample to DAC
    repeat(#50)        ;Wait
    NOP
    goto Begin_Cycle

**Function Configures McBSP0**

McBSP0_init:

    mmr(McBSP0_SPSA) = #SPCR1 ;Reset McBSP0 Receiver
    
```

```

A=mmr(McBSP0_SPSD)          ;
A = #0FFFEh & A              ;RRST*=0
mmr(McBSP0_SPSD) =A

repeat(#5)
NOP

mmr(McBSP0_SPSA) = #SPCR2    ;Reset McBSP0 Receiver
A=mmr(McBSP0_SPSD)
A= A & #0FF7Fh                ;FRST*=0
A= A & #0FFBFh                ;GRST*=0
A= A & #0FFFEh                ;XRST*=0

A=A|#0200h                    ;enable free running mode
mmr(McBSP0_SPSD) = A          ;

repeat(#5)
NOP

mmr(McBSP0_SPSA) = #PCR      ;FSXM & CLKXM output pin driven
mmr(McBSP0_SPSD) = #0A00h    ;by sample-rate generator
                                ;FSRM & CLKRM input pins
                                ;driven by external source

repeat(#6)
NOP

mmr(McBSP0_SPSA) = #RCR1     ;
mmr(McBSP0_SPSD) = #0040h    ;One 16-bit Word per frame
repeat(#6)
NOP
mmr(McBSP0_SPSA) = #RCR2     ;One 16-bit Word per frame
mmr(McBSP0_SPSD) = #0041h    ;one bit receive bit data delay

repeat(#6)
NOP

mmr(McBSP0_SPSA) = #XCR1     ;
mmr(McBSP0_SPSD) = #0040h    ;One 16-bit Word per frame
repeat(#6)
NOP
mmr(McBSP0_SPSA) = #XCR2     ;One 16-bit Word per frame
mmr(McBSP0_SPSD) = #0041h    ;one bit receive bit data delay

repeat(#6)
NOP
mmr(McBSP0_SPSA) = #SRGR1    ;FS width=one clock
A=#0009h
mmr(McBSP0_SPSD) = A          ;SRG clock divider=9 (CPU Clock /1+CLKGDV)=10Mhz

repeat(#6)

```

```

NOP

mmr(McBSP0_SPSA) = #SRGR2    ;SRG clock derived from CPU clock
mmr(McBSP0_SPSD) = #2000h    ;FSX due to transfer DXR->XSR

repeat(#6)
    NOP

mmr(McBSP0_SPSA) = #SPCR2    ;
A=mmr(McBSP0_SPSD)
A= A | #0040h                ;GRST*=1
mmr(McBSP0_SPSD) = A        ;Frame Generator and sample
                             ;sample-rate generator enabled

repeat(#6)
    NOP

A= A | #0001h                ;XRST*=1
mmr(McBSP0_SPSD) = A        ;Transmitter enabled

repeat(#6)
    NOP

mmr(McBSP0_SPSA) = #SPCR1    ;RRST*=1
A= mmr(McBSP0_SPSD)
A= A | #0001h
mmr(McBSP0_SPSD) = A        ;Receiver enabled

repeat(#6)
    NOP

RETURN

**Function writes to Transmitter and Read from Receiver of McBSP0**
**Data to be written out must be stored in Accum. A,                **
**Data received is stored in Accumulator B                          **
McBSP0WriteRead:          ;input in A
    PUSH (AR2)            ;Save Registers
    PUSH (AR1)
    PUSH (AR0)
Wait_On_Transmitter:
    DP=#0
mmr(McBSP0_SPSA) = #SPCR2    ;Check to see if transmitter is ready to send
B=mmr(McBSP0_SPSD)          ;new data.

B=B & #0002h;
AR1=B
nop
nop
AR0=#2
nop
nop
TC      = (AR0 == AR1)
    if (NTC) goto Wait_On_Transmitter ;If transmitter is full wait until it is not.
    
```

```
    AR2=#McBSP0_DXR1
    nop
    *AR2 = A                ;Transmit contents of accumulator A
Wait_On_Receiver:
    DP=#0
    mmr(McBSP0_SPSA) = #SPCR1    ;
    B=mmr(McBSP0_SPSD)          ;Check to see if Receiver is FULL with Data
    AR0=#2
    B=B & #0002h;
    AR1=B
    nop
    nop
    TC = (AR0 == AR1)
    if (NTC) goto Wait_On_Receiver ;If receiver is not ready with new data, then wait.
    DP=#AD_DP
    AR2=#McBSP0_DRR1
    NOP
    B=*AR2                ;Read out receiver
    AR0 =POP()            ;read sample into accumulator B
    AR1 =POP()
    AR2 =POP()
    RETURN
.end
```

## Appendix B C5402 Registers

```

*=====
*  FILENAME:  Reg.h
*
*  TMS320VC5402 & 5402 DSK memory mapped reg definition
*
*=====
*----  include the 54xx register map defined under .mmreg directive ----
      .mmregs
*-----
*----- TIMER2 Registers -----
TIM1      .set    0030h    ; Timer1
PRD1      .set    0031h    ; Timer1 Period Reg
TCR1      .set    0032h    ; Timer1 Ctrl Reg
*-----
*----- McBSP0 Registers -----
McBSP0_DRR2x .set    0020h    ; McBSP0 Data Rx Reg2
McBSP0_DRR1  .set    0021h    ; McBSP0 Data Rx Reg1
McBSP0_DXR2  .set    0022h    ; McBSP0 Data Tx Reg2
McBSP0_DXR1  .set    0023h    ; McBSP0 Data Tx Reg1
McBSP0_SPSA  .set    0038h    ; McBSP0 Sub Bank Addr Reg
McBSP0_SPSD  .set    0039h    ; McBSP0 Sub Bank Data Reg
*-----
*----- McBSP1 Registers -----
McBSP1_DRR2  .set    0040h    ; McBSP1 Data Rx Reg2
McBSP1_DRR1  .set    0041h    ; McBSP1 Data Rx Reg1
McBSP1_DXR2  .set    0042h    ; McBSP1 Data Tx Reg2
McBSP1_DXR1  .set    0043h    ; McBSP1 Data Tx Reg1
McBSP1_SPSA  .set    0048h    ; McBSP1 Sub Bank Addr Reg
McBSP1_SPSD  .set    0049h    ; McBSP1 Sub Bank Data Reg
*-----
*----- McBSP0 & McBSP1 Sub-Bank Addressed Registers -----

SPCR1      .set    0000h    ; McBSP Ser Port Ctrl Reg1
SPCR2      .set    0001h    ; McBSP Ser Port Ctrl Reg2
RCR1       .set    0002h    ; McBSP Rx Ctrl Reg1
RCR2       .set    0003h    ; McBSP Rx Ctrl Reg2
XCR1       .set    0004h    ; McBSP Tx Ctrl Reg1
XCR2       .set    0005h    ; McBSP Tx Ctrl Reg2
SRGR1      .set    0006h    ; McBSP Sample-rate Gen Reg1
SRGR2      .set    0007h    ; McBSP Sample-rate Gen Reg2
MCR1       .set    0008h    ; McBSP Multichan Reg1
MCR2       .set    0009h    ; McBSP Multichan Reg2
RCERA      .set    000Ah    ; McBSP Rx Chan Enable Reg Partition A

```

```

RCERB      .set      000Bh      ; McBSP Rx Chan Enable Reg Partition B
XCERA      .set      000Ch      ; McBSP Tx Chan Enable Reg Partition A
XCERB      .set      000Dh      ; McBSP Tx Chan Enable Reg Partition B
PCR        .set      000Eh      ; McBSP Pin Ctrl Reg
*----- General Purpose I/O Registers -----
GPIOCR     .set      003Ch      ; GP I/O Pins Control Reg
GPIOSR     .set      003Dh      ; GP I/O Pins Status Reg

```

```

*----- Onboard I/O Memory Mapped Register -----
DSP_CPLD_CNTL1      .set      0000h    ; Control Reg1
DSP_CPLD_STAT       .set      0001h    ; Status Reg
DSP_CPLD_DMCNTL     .set           0002h    ; Data Memory Control Reg
DSP_CPLD_DBIO       .set      0003h    ; Daughter Brd / GPIO Reg
DSP_CPLD_CNTL2     .set      0004h    ; Control Reg2
DSP_CPLD_SEM0       .set      0005h    ; Semaphore 0
DSP_CPLD_SEM1       .set      0006h    ; Semaphore 1
    
```

## Appendix C C5402 Vector Table

```

*****
*      FILENAME: c5402VEC.ASM
*      This routine initializes the 54xxDSK Interrupt vector table.
*****
        .title  "54xDSKPLUS Vector Table Initialization"
        .global main
        .algebraic
        .sect   ".vectors"

RESET:   dgoto  main           ; RESET vector
        nop
        nop
NMI:     dgoto  NMI           ;Nonmaskable Interrupt
        nop
        nop
*****
*      S/W Interrupts
*****
SINT17   return_enable       ;Software Interrupt #17
        nop
        nop
        nop
SINT18   return_enable       ;Software Interrupt #18
        nop
        nop
        nop
SINT19   return_enable       ;Software Interrupt #19
        nop
        nop
        nop
SINT20   return_enable       ;Software Interrupt #20
        nop
        nop
        nop
SINT21   return_enable       ;Software Interrupt #21
        nop
        nop
        nop
SINT22   return_enable       ;Software Interrupt #22

```

```

        nop
        nop
        nop
SINT23  return_enable      ;Software Interrupt #23
        nop
        nop
        nop
SINT24  return_enable      ;Software Interrupt #24
        nop
        nop
        nop
SINT25  return_enable      ;Software Interrupt #25
        nop
        nop
        nop
SINT26  return_enable      ;Software Interrupt #26
        nop
        nop
        nop
SINT27  return_enable      ;Software Interrupt #27
        nop
        nop
        nop
SINT28  return_enable      ;Software Interrupt #28
        nop
        nop
        nop
SINT29  return_enable      ;Software Interrupt #29
        nop
        nop
        nop
SINT30  return_enable      ;Software Interrupt #30
        nop
        nop
        nop
*****
*      Rest of the Interrupts
*****
INT0:  return_enable      ;External user interrupt #0
        nop
    
```

```

    nop
    nop
INT1: return_enable    ;External user interrupt #1
    nop
    nop
    nop
INT2: return_enable    ;External user interrupt #2
    nop                ;Only 2 NOPS
    nop
    nop
TINT: return_enable    ;Timer0 Interrupt
    nop
    nop
    nop
BRINT0: return_enable ;McBSP#0 Receiver Interrupt
    nop
    nop
    nop
BXINT0: return_enable ;McBSP#0 Transmit Interrupt
    nop
    nop
    nop
DMACO:  return_enable ;DMA Channel 0 interrupt
    nop
    nop
    nop
TINT1:  return_enable ;Timer1 Interrupt(default) or DMA channel 1 interrupt
    nop
    nop
    nop
INT3: return_enable    ;External user interrupt #3
    nop                ;only TWO NOPS
    nop
    nop
HPINT:  return_enable ;HPI Interrupt
    nop
    nop
    nop
BRINT1: return_enable ;McBSP#1 Receive Interrupt (Default) or DMA Channel 2
interrupt
  
```

```
    nop
    nop
    nop
BXINT1:  return_enable  ;McBSP#1 transmit interrupt (Default) or DMA Channel 3
          ;interrupt

    nop
    nop
    nop
DMAC4:   return_enable  ;DMA channel 4 interrupt

    nop
    nop
    nop
DMAC5:   return_enable  ;DMA channel 5 interrupt

    nop
    nop
    nop
```

## Appendix D C5402 Memory Mapping

```

/*****/
/*   TMS320C54x DSK Plus Linker Command File
/* 16K words on chip DARAM is shared by Prog
/* and Data sections
/*****/
MEMORY
{
    PAGE 0   :   /* Pgm space   */
    VECS     :   origin = 0080h, length = 0080h /* vector table space   */
    PROG     :   origin = 0100h, length = 1EFFh /* Pgm mem space       */

    PAGE 1   :   /* Data space */
    DAT0     :   origin = 0060h, length = 0020h /* Scratch pad mem space */
    DAT1     :   origin = 2000h, length = 1C00h /* 7K words for Data     */
    STK      :   origin = 3C00h, length = 0400h /* 1K words for Stack    */
}
SECTIONS
{
    .vectors: {} > VECS     PAGE 0   /* Interrupt Vector table */
    .coeffs: {} >  PROG     PAGE 0   /*                          */
    .data:   {} >  PROG     PAGE 0   /*                          */
    .text:   {} >  PROG     PAGE 0   /* Program code goes here  */
    .varibl: {} >  DAT1     PAGE 1   /* Uninitialized variables  */
    .bss:    {} >  DAT1     PAGE 1   /* Uninitialized variables  */
    .stack:  {} >  STK      PAGE 1   /* Software stack section   */
    .cinit:  {} >  DAT1     PAGE 1   /*                          */
}

```

## 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.

### Mailing Address:

Texas Instruments  
Post Office Box 655303  
Dallas, Texas 75265