A methodology of interfacing serial A-to-D converters to DSPs

By Thomas Kugelstadt
Application Manager

Introduction
Designers of DSP systems often have to rewrite their interface software when a desired increase in system performance requires the replacement of the current A-to-D converter with a device of higher speed or resolution. This article describes a method for interfacing various types of serial ADCs to the standard serial port of a DSP (TMS320C50) while keeping the software modifications at a minimum. It concludes with the introduction of “C-callable assembler routines” provided by TI’s application staff that relieve the experienced C-programmer from performing tedious assembler studies.

Serial analog-to-digital converters
Table 1 lists a range of serial ADCs that interface directly to DSPs without additional control logic. All these devices have the following in common:
- the conversion is performed by successive approximation based on charge redistribution,
- a conversion is initiated by an external trigger signal, and
- the data format for the transfer between DSP and ADC is 16 bits.

For purposes of describing interface timing the devices in Table 1 are grouped in two categories:
- on-the-fly converters that perform a conversion while exchanging data with the host, and
- sequential converters that execute a conversion following a data transfer.

Figure 1 shows a typical interface between a DSP and a serial ADC. The DSP general-purpose I/O XF signal activates the ADC. The control and data signals of the DSP serial port manage the transfer of data. The additional EOC/INT output, only available on sequentially converting ADCs, signals the end of a conversion to the DSP, indicating that a new transfer can begin.

Before the timing sequence of a data transfer is discussed, the reader should understand the internal operation of the standard serial interface.

Standard serial interface
TI provides a variety of DSPs with different types of serial interfaces including:
- buffered serial port (BSP) with auto-buffering unit module (AMU),
- buffered serial interfaces with internal FIFO, and
- time-division-multiplex (TDM) serial interfaces, particularly useful for telecom applications.

The standard serial interface is the focus of this discussion because it is used far more often than the BSP or the TDM serial port. The serial port interfaces to data converters via the following six control lines:

- CLKX Transmit clock input or output. This signal clocks data from the transmit shift register (XSR) to the DX pin. The serial port can be configured for internal clock generation or to accept an external clock. If the port is configured to generate the

Continued on next page

### Table 1. Family of serial A-to-D converters

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>CONVERSION</th>
<th>SUPPLY (V)</th>
<th>RESOLUTION (bits)</th>
<th>INPUT CHANNEL</th>
<th>CONVERSION RATE (MSPS)</th>
<th>POWER DOWN</th>
<th>SWEEP MODE</th>
<th>FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLV1570</td>
<td>On-the-fly</td>
<td>2.7 – 5.5</td>
<td>10</td>
<td>1</td>
<td>1.25</td>
<td>Auto</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>TLV1572</td>
<td>On-the-fly</td>
<td>2.7 – 5.5</td>
<td>10</td>
<td>8</td>
<td>1.25</td>
<td>Auto</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>TLV1544</td>
<td>Sequential</td>
<td>2.7 – 5.5</td>
<td>10</td>
<td>4</td>
<td>0.1</td>
<td>Prog.</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>TLV1548</td>
<td>Sequential</td>
<td>2.7 – 5.5</td>
<td>10</td>
<td>8</td>
<td>0.1</td>
<td>Prog.</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>TLV2544</td>
<td>Sequential</td>
<td>2.7 – 5.5</td>
<td>12</td>
<td>4</td>
<td>0.2</td>
<td>Prog.</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>TLV2548</td>
<td>Sequential</td>
<td>2.7 – 5.5</td>
<td>12</td>
<td>8</td>
<td>0.2</td>
<td>Prog.</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>TLC2554</td>
<td>Sequential</td>
<td>5.0</td>
<td>12</td>
<td>4</td>
<td>0.4</td>
<td>Prog.</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>TLC2558</td>
<td>Sequential</td>
<td>5.0</td>
<td>12</td>
<td>8</td>
<td>0.4</td>
<td>Prog.</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>
data clock on-chip, CLKX becomes an output, providing the data clock for the serial interface. If the port is configured to accept an external clock, CLKX changes to an input, receiving the external clock signal.

FSX  *Transmit frame synchronization input or output*. FSX indicates the start of a transmission. The serial port can be configured for internal frame-sync generation or to accept an external frame-sync signal. If the port is configured to generate the frame sync pulse on-chip, FSX becomes an output. If the port is configured to accept an external frame sync pulse, this pin becomes an input.

DX  *Serial data transmit*. DX transmits the actual data from the transmit shift register (XSR).

CLKR  *Receive clock input*. CLKR always receives an external clock for clocking the data from the DR pin into the receive shift register (RSR).

FSR  *Receive frame synchronization input*. FSR always receives an external frame sync pulse to initiate the reception of data at the beginning of a frame.

DR  *Serial data receive*. DR receives the actual data which are clocked into the receive shift register (RSR).

Serial port general operation

In the transmit direction the CPU initiates a data transfer by writing transmit data to the DXR. Then the data are copied from the DXR into the XSR and clocked out to the DX output. Upon the completion of a DXR-to-XSR copy, a transmit interrupt, XINT, is generated. This interrupt signals the CPU that new data can be written into the DXR.

In the receive direction the incoming data are clocked into RSR and then copied into DRR. Upon the completion of the data copy, a receive interrupt, RINT, is generated. This interrupt signals the CPU that new data are available in DRR. The CPU needs to read these data, while the next receive frame is clocked into the RSR.

Serial port configuration

Before a data transfer can be executed, the serial port needs to be configured via the serial port control register, SPC. Figure 3 shows the 16-bit memory-mapped SPC of the TMS320C50DSP (R = read-only bits, R/W = read/write bits).

Out of the 16 bits of the SPC, only six R/W bits (shaded bits in Figure 3) are used to configure the serial port. The remaining non-shaded R/W bits, such as the emulation bits FREE and SOFT, and the digital loop back bit, DLB, are set to zero (FREE = SOFT = DLB = 0).
The following example shows a typical serial port configuration when interfacing to a serial A-to-D converter from Texas Instruments.

- For standard serial port operation, all non-shaded R/W bits are set to zero (Free = Soft = DLB = 0).
- During the SPC configuration the serial port transmitter and receiver need to be disabled by setting the respective reset bits to zero (/XRST = /RRST = 0).
- Assuming that no external clock and frame sync generator are used, both signals need to be generated on-chip, thus requiring that TXM and MCM be set to one (TXM = MCM = 1).
- To comply with the 16-bit data format of the A-to-D converter, FO is set to zero (FO = 0).
- As mentioned previously, all serial ADCs require a trigger signal for each data transfer plus subsequent conversion. As will be shown in the following section, this trigger signal is already available through the FS-pulse of the serial port. For this purpose the serial port needs to be configured for burst mode operation by setting FSM to one (FSM = 1).

Figure 4 shows the resulting binary pattern with the corresponding hex code that is loaded into the SPC.

The assembler instruction << SPLK #0038h , SPC >> commands the CPU to write the content 38h into the serial control register, SPC. The CPU then activates the serial port via a second instruction << SPLK #00F8h , SPC >> by setting the /XRST and /RRST bit to one.

**Transmit and receive operations in burst mode**

The serial port can be programmed for continuous mode or burst mode operation. Both modes indicate the beginning of a data frame via a frame-sync pulse at FS. In continuous mode, only one FS-pulse is needed to indicate the beginning of a number of consecutive data frames. In burst mode, an FS-pulse indicates the start of one data frame (16 SCLK cycles) only, thereby providing the required trigger signal for an A-to-D conversion.

Therefore, the serial port needs to operate in burst mode. Figure 5 shows the interface timing of the serial port in burst mode operation.

The data clock has the two designators CLKX and CLKR. The data clock input of the serial port receiver. The same is identical to the one of the serial port transmitter. Figure 1 shows that when the serial port is configured for on-chip clock generation (MCM = 1), CLKX is not only connected to the clock input of the ADC but also fed back into the data clock input of the serial port receiver. The same is valid for the frame-sync signals, FSX and FSR.

A data transfer is initiated by the CPU writing transmit data (i.e., ADC configuration data) into the transmit data register, DXR. With the second rising edge of CLKX, the content of DXR is copied into the transmit shift register, DXR. At this time, a frame-sync pulse and a transmit interrupt, XINT, are generated. With the first rising edge of CLKX after FSX has gone low, transmit data are shifted out to the DX-pin while receive data are clocked into the receive shift register, RSR. At this time, a frame-sync pulse and a transmit interrupt, XINT, are generated. With the first rising edge of CLKX after FSX has gone low, transmit data are shifted out to the DX-pin while receive data are clocked into the receive shift register, RSR, via the DR-pin. Both data streams, in transmit and receive, start with the most significant bit, MSB. While data change with the rising edge of CLKX, the falling edge of CLKX latches the transmit data into the ADC and the receive data into RSR. With the 16th falling edge of CLKX (after FSX has gone low), the receive shift register, RSR, is full. The content of RSR is copied into the data receive register, DDR, and a receive interrupt, RINT, is generated. The falling edge of the 16th CLKX cycle completes an entire data transfer, both in receive and transmit. Any further data transfer needs to be initiated with a CPU reload of DXR.

The serial port interrupts, XINT and RINT (shaded gray in Figures 5 and 6), represent internal signals, which are available to the CPU only. If either one of these interrupts

---

**Table 2. Important SPC bits for an ADC/DSP interface**

<table>
<thead>
<tr>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>/XRST/RRST</td>
<td>The Transmit and Receive reset signals activate and deactivate the transmitter and receiver of the serial port. /XRST, /RRST = 1, transmitter and receiver are active /XRST, /RRST = 0, activity halts</td>
</tr>
<tr>
<td>TXM</td>
<td>The Transmit Mode bit specifies the source for FSX-pulse generation. TXM = 1, FSX is generated on-chip and synchronized to CLKX TXM = 0, FSX needs to be applied from external source</td>
</tr>
<tr>
<td>MCM</td>
<td>The Clock Mode bit specifies the clock source for CLKX. MCM = 1, on-chip clock source is used MCM = 0, external clock source is chosen</td>
</tr>
<tr>
<td>FSM</td>
<td>The Frame Synch Mode bit specifies when a frame sync pulse is needed. FSM = 1, Burst Mode is selected (an FS-pulse is used for each word) FSM = 0, Continuous Mode is selected (only one start pulse is required)</td>
</tr>
<tr>
<td>FO</td>
<td>The Format bit specifies the word length of the transmitter and receiver. FO = 0, word length is 16-bit FO = 1, word length is 8-bit</td>
</tr>
</tbody>
</table>

---

**Figure 4. Serial port configuration code**

<table>
<thead>
<tr>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREE</td>
<td>SOFT</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure 5. Transmit and receive operation in burst mode

Figure 6. Burst mode operation at maximum packet frequency
occurs, the program execution is directed to a transmit interrupt service routine, XINT-ISR, or a receive interrupt service routine, RINT-ISR. Within those ISRs, the CPU can execute serial port supporting tasks while a data transfer over the serial interface is still in progress.

For example, an XINT-ISR is often used to reload DXR after the old DXR content is copied into XSR. This ensures that the next DXR-to-XSR copy happens as soon as XSR is empty and a second data transmit can follow immediately. However, as will be shown in the next section, when interfacing to sequentially operating ADCs, XINT needs to be disabled. A RINT-ISR is used to read the data receive register, DRR, and to save the content into data memory. When interfacing to on-the-fly converters, the RINT-ISR can also include the initiation of the next data transfer to maximize the data throughput.

A specific case of the burst mode is the operation at maximum packet frequency, shown in Figure 6.

At maximum packet frequency, the data bits in consecutive packets are transmitted contiguously with no inactivity between bits. This is achieved by reloading DXR during the first XINT-ISR. A second instruction disables XINT for all further data frames. A XINT-ISR executing these two instructions takes approximately five CLKX cycles to complete. With the rising edge of the 16th CLKX cycle, the last transmit bit is shifted out of XSR, and the latest DXR content is copied into XSR. At this time a new frame sync pulse, which overlaps the last transmit bit of the previous frame, is generated to start a new data transfer.

With the 16th falling edge of CLKX, half a clock cycle later, the last receive bit is clocked into the RSR. The subsequent RSR-to-DRR copy generates a receive interrupt. During the following RINT-ISR, the CPU reloads the transmit data register, DXR. It then reads the receive data register, DRR, and saves its content into data memory. While the CPU is executing the RINT-ISR, the serial port continues to transfer data. From now on, only one interrupt service routine, the RINT-ISR, is used to save the receive data into memory, as well as to initiate all further data transfers. Care must be taken when adding further instructions to the RINT-ISR. Since the data stream needs to be contiguous, the length of the ISR must not exceed 15 CLKX cycles.

Interfacing to sequential ADCs

The sequential A-to-D converters in Table 1 operate in two phases. In phase 1, the ADC simultaneously receives configuration data from the DSP and transmits conversion results to the DSP. During this data transfer, the ADC is configured for the desired operation mode and starts sampling the analog input channel. Phase 2 represents the actual analog-to-digital conversion, followed by an interrupt signal once a conversion is complete. The advantage of the sequential operation is that the sampling period can be extended by a factor of 2 without affecting the conversion time. This is particularly useful for sampling high-impedance input sources. A long sampling time allows the ADC internal switched capacitors to be charged longer, thus helping the analog input signal sampled to settle within a 0.5-LSB accuracy.

Figure 7 shows the timing diagram of the TLV2544 ADC in single-shot mode. Because a conversion needs to be completed before another one can be started, a contiguous flow of data bits is not possible, which is true for all sequentially operating ADCs.

Note that most ADCs provide a programmable EOC/INT pin to signal the end of a conversion. The following discussion assumes that the EOC/INT pin is programmed to use the /INT pulse to indicate the end of a conversion. The two gray-shaded signals, EOC and RINT, serve demonstration purposes only.

EOC presents the internal conversion time of the A-to-D converter, while RINT demonstrates the occurrence of the DSP internal receive interrupt.

Continued on next page
The data acquisition process starts with the DSP providing the data clock, SCLK, for the serial interface. The DSP then activates the ADC by applying a logic low via the external flag output, XF, to the /CS input of the A-to-D converter. With the falling edge of /CS, the data output of the ADC leaves the high-impedance state and provides a random logic value at DOUT.

After the ADC is enabled, each data transfer is executed by the following five steps:

1. In the main routine, the DSP initiates a data transfer by writing ADC configuration data into the DXR of the serial port. On the second rising edge of SCLK following the DXR-write, a frame sync pulse, FS, is generated.

2. With the falling edge of FS, the serial port transmits configuration data to the ADC via DIN. The first four bits represent the actual configuration data, while the remaining 12 bits are ignored by the ADC. Simultaneously, the ADC transmits the conversion results via DOUT to the DSP. To comply to the 16-bit frame format, the results of a 10-bit converter is followed by six zero bits. A 12-bit conversion result requires only four trailing zeros.

3. With the 16th falling edge of SCLK, the DX output, and with it DIN, goes into high-impedance, while DOUT stays low till the next data transfer. The A-to-D converter starts the conversion process and the generated receive interrupt of the serial port leads the CPU to enter the RINT-ISR. During this interrupt routine, the

CPU reads the receive data from the serial port and saves it into data memory.

4. Upon the completion of the RINT-ISR the CPU idles until an unmasked interrupt occurs. The ADC completes the conversion and provides an interrupt pulse via its /INT output to one of the DSP external interrupt inputs, /INT1 – /INT3.

5. The CPU now enters the INTx-ISR, reloading the DXR with ADC configuration data (i.e., the channel number). The new DXR-write initiates the next data transfer sequence, and Steps 1–4 are repeated until all samples have been acquired.

Figure 8 shows the basic program flow that supports the above interface timing.

**Interface program for sequential ADCs**

The main routine starts with defining the following variables:

- the number of samples to be acquired,
- the memory start address for the acquired data, and
- the ADC configuration words such as the conversion mode, the sampling mode, and the channel numbers.

The DSP configuration disables all interrupts, specifies the wait-states to zero, and defines the program memory range for the data acquisition program. Then the serial port is configured for burst mode operation and on-chip data clock and frame-sync generation.

To configure the A-to-D converter, the DSP takes the XF-output low and enables the ADC via the /CS-input. The CPU loads the DXR with ADC configuration data, causing the first data transfer to start. Then the CPU is set into idle mode, waiting for interrupts to occur.

Once an ISR has been completed, the CPU checks for program completion. As long as the END-flag is zero, the CPU returns to idle mode. If the END-flag is one, the DSP disables the ADC by taking XF high, thus completing the main routine.

**RINT-ISR**

During the RINT-ISR the CPU reads the receive data in the serial port, saves the data into memory, and returns to the main routine.

**INTx-ISR**

The INTx-ISR initiates all further data transfers. As long as more samples are to be acquired, the CPU continuously reloads the DXR to start a new conversion.

Once all samples have been acquired, the END-flag is set to one and program control returns to the main routine.
16-bit configuration word to the ADC. In receive, the ADC provides 16-bit output data, consisting of four preceding zeros and a 12-bit conversion result. Meanwhile the XINT-ISR reloads DXR and disables XINT for all further data frames.

2. While the data transfer continues, the CPU idles, waiting for a receive interrupt to occur.

3. With the rising edge of the 16th SCLK cycle, a new FS-pulse starts the next data frame. Upon the falling edge of the 16th SCLK cycle, a RINT is generated that forces the CPU to leave the idle mode and enter the receive interrupt service routine. Within the RINT-ISR, the CPU reads the DRR and saves the received data into memory. It then writes a new configuration word into DXR to prepare for the next data frame. After completing the RINT-ISR, the CPU returns to idle mode. Then Steps 2 and 3 are repeated until all samples have been acquired.

Figure 10 shows the timing diagram of the 12-bit, single-channel ADC, TLV1572. The interface timing is similar to that of the TLV1570, except that the output data of the continued on next page
Continued from previous page

TLV1572 represent the results of the current conversion, while the output data of the TLV1570 are the results of the conversion executed in the previous data frame.

The TLV1572 is not configurable and therefore provides no DIN terminal. The DSP still needs to perform a dummy write of the DXR (i.e., with random data) to generate the required FS-pulse, which indicates the start of a data frame and triggers the A-to-D conversion.

Figure 11 shows the basic program flow that supports the above interface timing.

C-callable interface routines

The previous flowcharts describe the structure of the interface routines written in assembler. These assembler routines are used to investigate and to optimize the interface timing between newly released data converters and DSPs. However, with the majority of DSP programmers using C rather than assembler, TI’s application team for analog products has made its assembler routines C-callable.

Figure 12 gives an example of a C-callable assembler routine for the 8-channel, 12-bit ADC, TLV2548.

In addition, the assembler routine received a save-context and a restore-context task.

- **Save-context** saves all pointers and registers, such as frame-pointer, stack-pointer, status and auxiliary registers, which have been used previously in the C-program.
- **Restore-context** restores the original content of these registers before returning to the C-program.

After the variable definition in C, a call instruction starts the assembler routine. All previously used registers are saved, and the actual data acquisition begins. Once the user-defined number of samples has been received, all pointers and registers are restored. The CPU exits the assembler routine and returns to the C-program. Using C-callable interface relieves the programmer from the tedious task of studying device-specific assembler code.

---

**Figure 11. Interface program for on-the-fly ADCs**

---

**Main routine**

The main routine is almost the same as the one in Figure 8, except that the internal transmit interrupt, XINT, is used instead of the external interrupt, INTx.

The main routine starts with defining the number of samples to be acquired, the memory start address for the acquired data, and the ADC configuration words. In the case of a non-configurable device, there are no configuration words. Instead, any random code can be written to DXR to start a data transfer.

The main routine continues with configuring the DSP and the serial port.

Then the two serial port interrupts, RINT and XINT, are enabled, and the END-flag is set to zero.

To configure the ADC, the DSP takes the XF-output low and enables the device via the /CS- input. The CPU writes configuration data to DXR. The following data transfer configures the ADC, while the CPU goes into idle mode, waiting for interrupts to occur.

Once an ISR has been completed, the CPU checks for program completion. Depending on the status of END, the CPU either returns to idle mode or disables the ADC and completes the main routine.

**XINT-ISR**

With the start of the first data frame, a XINT is generated. The CPU enters the XINT_ISR and reloads DXR to ensure an immediate start of the second frame, once the first transfer is completed and XSR is empty.

Then the CPU disables XINT for all further transfers.

**RINT-ISR**

This RINT_ISR is the sum of the previous RINT- and INTx-ISRs for sequential converters. The CPU reads the receive data and saves it into memory.

As long as more samples are to be acquired, the CPU continuously reloads the DXR to start a new conversion. Once all samples have been acquired, the END-flag is set to one and program control returns to the main routine.
Figure 12. C-callable interface program

C2548.ASM

C2548:

* Save Context
  - Save Return Address, FP, SP, ST0, ST1, AR6, AR7, SWWSR

* Data Acquisition
  (Conversion & Transfer)

* Restore Context
  - Restore Return Address, FP, SP, ST0, ST1, AR6, AR7, SWWSR

* Return to C Program
  Ret

C2548.C

Main()
{
  /* select Vcc, Vcc = 5 or 3
     Vcc = 5;
  /* select channel */
     Channel = 0x2000;
  /* select samples */
     Samples = 0x1000;
  /* select memory address*/
     MemStart = 0x2000;
     ...

  /*Call Interface Program */
  C2548();

  /* remaining C program */
  ...
}

References
For more information related to this article, visit the TI Web site at www.ti.com/ and look for the following materials by entering the TI literature number into the quick-search box.

Document Title  TI Lit. #
1. Analog Applications (August 1999)  SLYT005
2. Characteristics, Operation, and Use of the TLV157XEVM  SLA0025
3. Interfacing the TLV1572 ADC to the TMS320C203 DSP  SLAA026B
4. TLV1570 EVM User's Guide  SLA0026
5. TLV1572 EVM User's Guide  SLA0027
6. Choosing an ADC and Op Amp for Minimum Offset  SLAA064
7. Interfacing the TLV1544 ADC to the TMS320C203 DSP  SLAA028A
8. Interfacing the TLV1544 ADC to the TMS320C50 DSP  SLAA025A
9. Switched-capacitor ADC Analog Input Calculations  SLAA036

Related Web sites
Get product data sheets at:
www.ti.com/sc/docs/products/analog/device.html
Replace device with tlc2554, tlc2558, tlv1544, tlv1548, tlv1570, tlv1572, tl2544, or tl2548
www.ti.com/sc/docs/products/dsp/tms320c50.html
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 beyond the parameters stated by TI for that product or service.

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 and cannot 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 products or services and is a unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

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

Products
Amplifiers amplifier.ti.com
Data Converters dataconverter.ti.com
DSP dsp.ti.com
Interface interface.ti.com
Logic logic.ti.com
Power Mgmt power.ti.com
Microcontrollers microcontroller.ti.com

Applications
Audio www.ti.com/audio
Automotive www.ti.com/automotive
Broadband www.ti.com/broadband
Digital control www.ti.com/digitalcontrol
Military www.ti.com/military
Optical Networking www.ti.com/opticalnetwork
Security www.ti.com/security
Telephony www.ti.com/telephony
Video & Imaging www.ti.com/video
Wireless www.ti.com/wireless

C011905
Safe Harbor Statement: This publication may contain forward-looking statements that involve a number of risks and uncertainties. These “forward-looking statements” are intended to qualify for the safe harbor from liability established by the Private Securities Litigation Reform Act of 1995. These forward-looking statements generally can be identified by phrases such as TI or its management “believes,” “expects,” “anticipates,” “foresees,” “forecasts,” “estimates” or other words or phrases of similar import. Similarly, such statements herein that describe the company’s products, business strategy, outlook, objectives, plans, intentions or goals also are forward-looking statements. All such forward-looking statements are subject to certain risks and uncertainties that could cause actual results to differ materially from those in forward-looking statements. Please refer to TI’s most recent Form 10-K for more information on the risks and uncertainties that could materially affect future results of operations. We disclaim any intention or obligation to update any forward-looking statements as a result of developments occurring after the date of this publication.

Trademarks: All trademarks are the property of their respective owners.

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

© 2005 Texas Instruments Incorporated