# Smallest DSP-compatible ADC provides simplest DSP interface

**By Joe Purvis**
*Application Specialist*

## Introduction

This article features TI's new TLV2541 12-bit data converter and highlights the ease with which analog signals can be converted.

## Family features

The TLV2541 is one of six related devices, presented in Table 1.

**Table 1. TLV25xx family features**

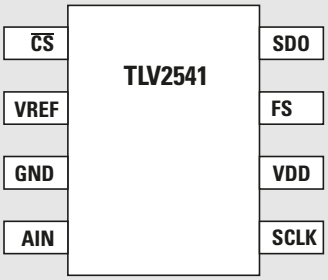
| DEVICE NUMBER | DESCRIPTION |
|---|---|
| TLV2541 | 1-channel, unipolar input, 200-KSPS*, 8-pin MSOP |
| TLV2542 | 2-channel, unipolar input, 200-KSPS*, 8-pin MSOP |
| TLV2545 | 1-channel pseudo-differential, 200-KSPS*, 8-pin MSOP |
| TLC2551 | 1-channel, unipolar input, 400-KSPS*, 8-pin MSOP |
| TLC2552 | 2-channel, unipolar input, 400-KSPS*, 8-pin MSOP |
| TLC2555 | 1-channel pseudo-differential, 400-KSPS*, 8-pin MSOP |

*DSP interface

### TLV2541

The pin-out for this device is shown in Figure 1. A significant feature is that the TLV2541 will support a typical digital signal processor (DSP) with no glue logic. It will also support a typical serial port protocol, such as can be found in 8- and 16-bit microprocessors. The device does not support interrupts or end-of-conversion (EOC) flags.

**Figure 1. TLV2541 device pinout**




### Timing diagram

The TLV2541 does not contain any user-configuration registers. Therefore, it is not possible to write to the device. After power-up, the ADC will power up in a known state.

There are two ways to trigger a conversion:

1. *Microprocessor mode*
   If the frame sync (FS) signal is **HIGH** during the falling edge of $\overline{CS}$, the ADC is considered to be in microprocessor mode. In this mode the sample/convert cycle occurs on every falling edge of $\overline{ADC\_CS}$. The user must ensure that $\overline{CS}$ remains active (LOW) for the entire sample/convert cycle (Figure 2).

   In this example, when $\overline{CS}$ is asserted LOW, the data from the previous sample/convert cycle is available on the falling edges of SCLK. The data received by the host system is B720 (h).

2. *DSP mode*
   If the FS signal is **LOW** during the falling edge of $\overline{CS}$, the ADC is said to be in DSP mode. In this mode, sample/convert cycles occur after FS has transitioned from HIGH to LOW (Figure 3).

   In this example, when $\overline{CS}$ is asserted LOW, the FS signal is LOW; therefore, the MSB of data output from the ADC will be held until the ADC detects FS↓. At that time, data from the previous sample/convert cycle is available on the falling edges of SCLK. The data received by the host system is 1640 (h).

   As can be seen in Figures 2 and 3, the serial data transmitted from the ADC in both instances follows a big endian data model (MSB to LSB).

**Continued on next page**

**Figure 2. Conversion trigger timing in microprocessor mode**

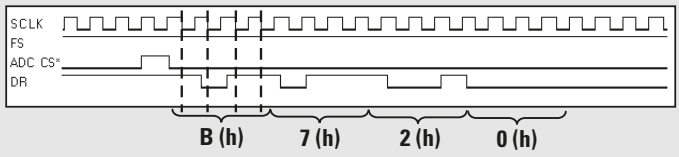
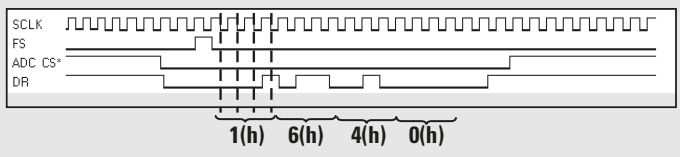



**Figure 3. Conversion trigger timing in DSP mode**



Continued from previous page

## Device evaluation

The device previously described is supported in an evaluation kit (Multi-Converter EVM). This kit contains:

- 0309 assembly, including socket support for all ADCs in this family, in addition to various digital-to-analog converters (DACs);
- samples of all available ADCs—TLV2541, TLV2542, TLV2545, TLC2551, TLC2552, and TLC2555; and
- samples of all available DACs—TLV5606, TLV5616, TLV5617A, TLV5618A, TLV5623, TLV5624, TLV5625, TLV5626, TLV5636, TLV5637, and TLV5638.

The EVM can be set up very quickly. Just follow these steps:

1. Apply power to the EVM.
2. Press the RESET button momentarily.
3. Press the START button momentarily.

Check that conversions are occurring by inspecting TP7 and TP20 (see Figure 4) via an oscilloscope. TP7 displays the input signal; TP20 displays the output signal.

## Programming the TLV2541 ADC

Programming the TLV2541 can be achieved with a minimum of effort, since there are no registers to set up. The converter begins a sample/convert cycle every time the $\overline{CS}$ signal falls (microprocessor mode) or FS falls (DSP mode).

For clarity, sample code for both the microprocessor and the DSP is presented. The code written in these examples is referenced to the EVM system previously discussed and includes a TLV5636 DAC as a partner device for the TLV2541.

### TLV2541 ADC flow

The goal of this code is to demonstrate the fundamental operation of the device. Several operations need to be completed successfully to achieve this goal, regardless of whether the platform is a DSP's serial port or a microprocessor's serial port:

- The host system (DSP or microprocessor) must be set up.
- The analog input signal must be sampled and converted.

**Figure 4. EVM functional diagram**

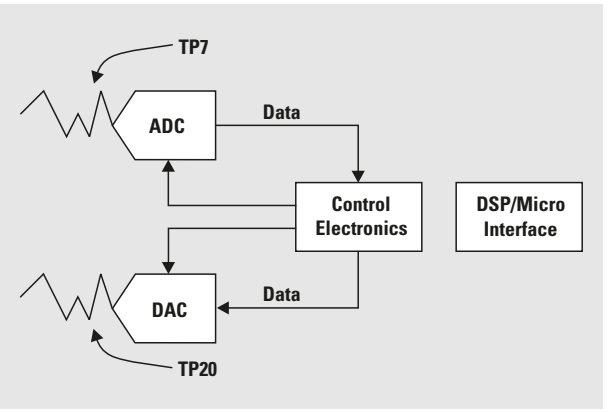
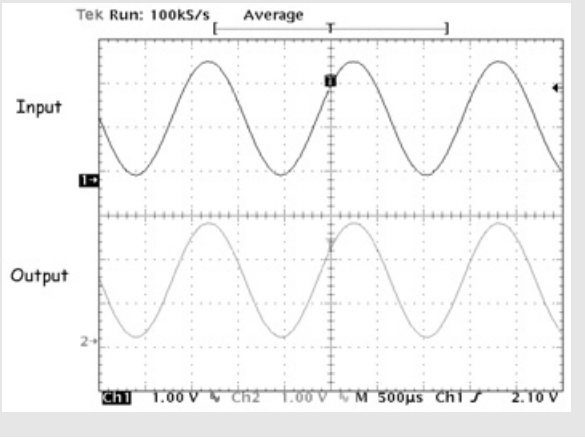



**Figure 5. TLV2541 input/output comparison**




- Converted data must be read into the host system as a serial bitstream.
- Some processing of the data word is necessary in preparation for writing the data to the DAC.
- The processed data is written to the DAC.

If these steps are completed successfully, the output signal from the on-board DAC will approximate the input signal from the ADC, as shown in Figure 5.

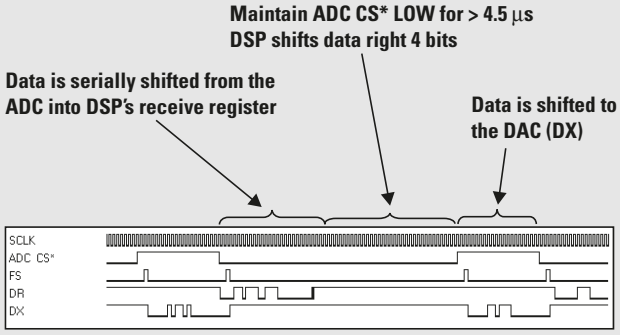
The only limit to this is the speed with which the DSP/microprocessor can read, process, and output the data.

Now that the general points have been discussed, example DSP code and microprocessor code are presented.

### DSP code

The following code example in assembly language shows serial port access of a Texas Instruments TMS320C31 DSP using the DSP Starter Kit (DSK). Figure 6 shows a typical example of the serial port data transfer timing.

**Figure 6. Serial port timing example**



### *Serial port setup*

The TMS320C31 contains one serial port. This article does not discuss the details of the serial port. See the references for more details. Table 2 lists the registers that must be set up and the data that should be written to the register in order to set up the serial port properly.

**Table 2. DSP serial port setup**

| REGISTER NAME | ADDRESS (HEX) | DATA (HEX) |
|---|---|---|
| Serial-port Global Control | 808040 | 0C140044 |
| FSX/DX/CLKX Port Control | 808042 | 00000111 |
| FSR/DR/CLKR Port Control | 808043 | 00000111 |
| R/X Timer Control | 808044 | 01CF |
| R/X Timer Counter | 808045 | 00100010 |
| R/X Timer Period | 808046 | 00000000 |
| Data Transmit | 808048 | Variable |
| Data Receive | 80804C | Variable |

### *Initiate a sample/convert*

Refer to Figure 3 for assistance.

- Drive $\overline{\text{ADC CS}}$ LOW.

```
ldi       2,iof
```

- Generate an FS signal.

```
sti       r0,@xdata
```

### *Read the data from the previous conversion into the host system*

- Because $\overline{\text{ADC CS}}$ was held LOW and FS was generated in the previous cycle, the serial data was automatically received by the DSP's receive register (DR) and assembled into a 16-bit word. This data can now be read into the DSP.

```
ldi       @rdata, r0
```

- The user must maintain $\overline{\text{ADC CS}}$ LOW for the complete sample/convert cycle. If $\overline{\text{ADC CS}}$ is de-asserted too early, the conversion cycle will be lost. There are a number of ways to achieve this delay; however, the simplest, without any knowledge of the DSP's serial port, is to include a decrement loop, timed for about 4.5 µs.

```
          ldi       0x10,r1
delay1    subi      1,r1
          bnz       delay1
```

The register is loaded with a value of 16 (10h) and is successively decremented until the register contains 0. The Z-flag in the DSP's status register is set, and the delay time is complete. It's clear that the number loaded is arbitrary and, depending upon the speed of the DSP, may not be optimal.

Maintaining $\overline{\text{ADC CS}}$ LOW for at least 4.5 µs achieves two objectives: It allows the ADC time to convert the analog input, and it also lets the ADC transmit the previously converted data to the DSP.

### *Data process*

It's clear from the data formats of the ADC (Figure 7) and the DAC (Figure 8) that the host system cannot simply write the ADC data to the DAC. The ADC data first must be shifted right by 4 bits as shown in Figure 9.

The control nibble sent to the DAC is defaulted to 0000(b). Other control nibbles may be written to the DAC; these are at the user's discretion. However, 0000(b) is interpreted by the DAC (TLV5636) as:

- Write to the DAC latch.
- DAC is in slow mode.
- Normal operation.

```
lsh       -4,r0
```

See Reference 2 for further information.

**Continued on next page**

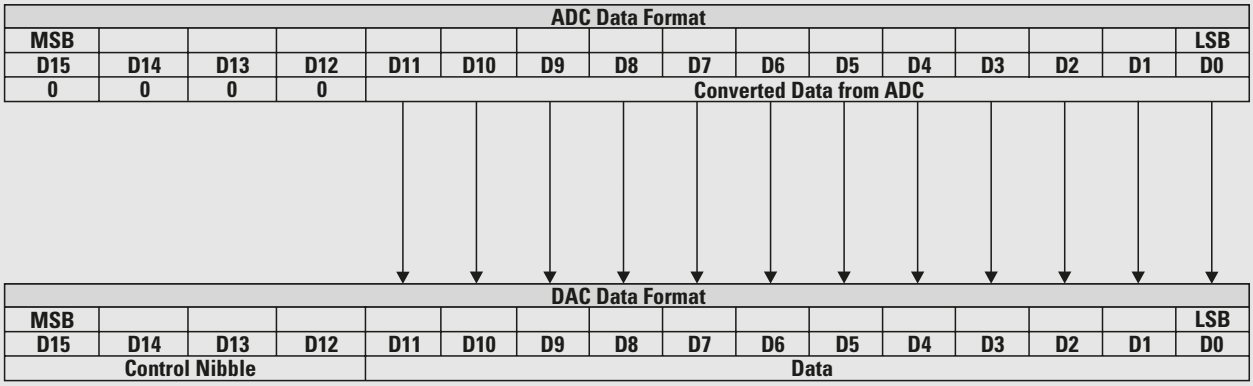
**Figure 7. Data received from the ADC**

| ADC Data Format | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSB | | | | | | | | | | | | | | | LSB |
| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Converted Data from ADC | | | | | | | | | | | | 0 | 0 | 0 | 0 |

**Figure 8. The DAC requires a 16-bit word that is composed of 2 parts**

| DAC Data Format | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSB | | | | | | | | | | | | | | | LSB |
| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Control Nibble | | | | Data | | | | | | | | | | | |

**Figure 9. ADC data is shifted right 4 bits before it is written to the DAC**




**Continued from previous page**

***Processed data is written to the DAC***

The data is written to the DAC bit by bit upon successive falling edges of SCLK. This process is initiated after $\overline{\text{DAC}}$ $\overline{\text{CS}}$ is asserted LOW and FS falls. After the 16th falling SCLK edge, the data in the DAC latch will be sent to the DAC for conversion.

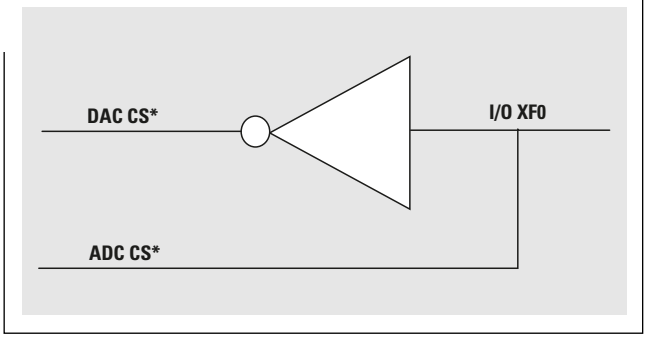
The ADC and the DAC share the same general-purpose I/O signal, I/O XF0. Selection of either device is mutually exclusive; i.e., if the ADC is selected the DAC is not selected, and if the DAC is selected the ADC is not selected. The ADC is selected by asserting I/O XF0 LOW; thus the DAC is selected by de-asserting I/O XF0 HIGH (see Figure 10).

```
ldi     6,iof
sti     r0, @xdata
```

Wait until all 16 bits have been transferred serially to the DAC.

```
        ldi     0x10,r1
delay2  subi    1,r1
        bnz     delay2
```

**Figure 10. Simple inverter allows ADC and DAC to share the same I/O signal**




The code portions just discussed can be assembled into the DSP program as given below:

## Program code sample

```
.start  "SPORT", 0x809802
       .sect "SPORT"
       .entry BEGIN

                                ;Serial Port 0 registers
sport       .set 0x808040       ;Serial Port 0 global control register
xpctrl      .set 0x808042       ;FSX/DX/CLKX port control
rpctrl      .set 0x808043       ;FSR/DR/CLKR port control
rxtctrl     .set 0x808044       ;R/X timer control register
rxtcnt      .set 0x808045       ;R/X timer counter register
rxtprd      .set 0x808046       ;R/X period register
xdata       .set 0x808048       ;R/X Data transmit register
rdata       .set 0x80804C       ;R/X Data receive register
t0_GO       .set 001cfh         ;Timer cfg to GO

sp0_cfg     .word   0x0c140044
s0_rxcntr   .word   0x00100010
s0_rxprd    .word   0x00000000
```

## Program code sample (Continued)

```
************************************************
* Main Program
************************************************
BEGIN    ldi    6,iof            ;Make sure ADC is not selected
         ldi    0x00000111,r0    ;Setup Serial Port:-
         sti    r0,@xpctrl           ;FSX is a serial port pin
                                     ;DX is a serial port pin
                                     ;CLKX is a serial port pin
         sti    r0,@rpctrl           ;FSR is a serial port pin
                                     ;DR is a serial port pin
                                     ;CLKR is a serial port pin
         ldi    @s0_rxcntr,r0    ;Counter value
         sti    r0,@rxtcnt       ;Load the counter with 15 for receive and transmit
         ldi    @s0_rxprd,r0     ;period value

         sti    r0,@rxtprd       ;Load the period with 3e for receive and transmit
         ldi    t0_GO,r0         ;Start timer0 config
         sti    r0,@rxtctrl      ;Start the counter
         ldi    @sp0_cfg,r0      ;0x0c140044 config for SP0 GCR
         sti    r0,@sport        ;configure sp0 GCR

*********************************************************************
* Sample and convert here
*********************************************************************

loop     ldi    2,iof            ;Assert CS* to select ADC
         sti    r0,@xdata        ;Generate an FS pulse

         ldi    @rdata,r0        ;Read ADC from the receive register

         ldi    0x10,r1          ;Wait for the ADC to complete a sample/convert cycle.
delay1   subi   1,r1             ;Wait for previous data to be received by DRR.
         bz     delay1           ;First conversion will be garbage.

          lsh   -4,r0            ;Shift the ADC data right 4 bits

         ldi    6,iof            ;Assert DAC CS*
         sti    r0,@xdata        ;Write data to the DAC

         ldi    0x10,r1          ;Wait for the serial data to be loaded into the DAC
delay2   subi   1,r1             ;before beginning another sample convert cycle.
         bz     delay2

         b      loop

         .end
```

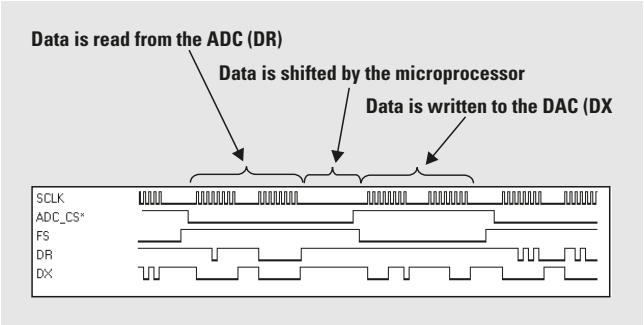
### Microprocessor code

The following code example in assembly language shows the SPI interface of the Motorola MC68HC912B32 microprocessor. General-purpose IO PS3 is used to generate frame sync (FS).

Since SPI is an 8-bit protocol, the communication to the ADC and DAC is achieved over two cycles. Figure 11 shows the converted data being read from the ADC and the processed data (4-bit shift) being written to the DAC.

**Continued on next page**

**Figure 11. Code transfer timing example**



**Continued from previous page**

***Serial port setup***

This article does not discuss the details of the serial port. See Reference 4 for more details. Table 3 lists the registers that must be set up and the data that should be written to the register in order to set up the serial port properly.

**Table 3. Microprocessor serial port setup**

| REGISTER NAME | ADDRESS (HEX) | DATA (HEX) |
|---|---|---|
| SPI Control Register 1 | 00D0 | 0054 |
| SPI Control Register 2 | 00D1 | 0000 |
| SPI Baud Rate Register | 00D2 | 0001 |
| SPI Status Register | 00D3 | 0080 (to clear) |
| SPI Data Register | 00D5 | Variable |
| Port S | 00D6 | Variable |
| Data Direction Register for Port S | 00D7 | 00EC |

***Initiate a sample/convert: Read data into host system***

Refer to Figure 2 for the timing diagram, FS = 1.

- Drive ADC CS* LOW.
  `MOVB #$80, PORTS`
- Generate 8 SCLKs to ADC by initiating a dummy data transfer.
  `MOVB #$00, SP0DR`
- Load the first 8 bits of sampled data in accumulator A.
  `LDAA SP0DR`
- Generate 8 SCLKs to ADC by initiating a dummy data transfer.
  `MOVB #$00, SP0DR`
- Load the second 8 bits of sampled data in accumulator B.
  `LDAB SP0DR`
- Store 16 bits of sampled data (accumulator D) in DATA.
  `STD DATA`

***Process data***

As in the DSP example, the control nibble sent to the DAC is defaulted to 0000(b). Other control nibbles may be written to the DAC; these are at the user's discretion. There are various ways to right shift the ADC data by 4 bits; in this example a simple loop is used.

- Load index register X with the value 4.
  `LDX #$0004`
- Loop four times, shifting right once on each iteration of the loop, then store the shifted values from accumulators A and B.

```
LOOP:   LSRD
        DBNE X,LOOP
        STAA UPPER_BYTE
        STAB LOWER_BYTE
```

***Processed data is written to the DAC***

Since the MC68HC912 operates on an 8-bit data word, the upper and lower bytes must be sent individually. This is done on a bit-by-bit basis, with MSB first. The ADC must be deselected, which will select the DAC as described in the DSP example given previously.

- Disable the ADC while enabling the DAC.
  `MOVB #$88, PORTS`
- Send DAC FS LOW by clearing the PS3 bit.
  `BCLR PORTS, #%00001000`
- Send upper and lower data bytes to the DAC.

```
MOVB UPPER_BYTE, SP0DR
BSR FLAG
MOVB LOWER_BYTE, SP0DR
BSR FLAG
```

- Branch back to sample routine.

The code portions just discussed can be assembled into the microprocessor program as given below:

## BRA sample program code

```
SP0CR1:      equ $D0      ;SPI 0 Control Register 1
SP0CR2:      equ $D1      ;SPI 0 Control Register 2
SP0BR:       equ $D2      ;SPI 0 Baud Rate Register
SP0SR:       equ $D3      ;SPI 0 Status Register
SP0DR:       equ $D5      ;SPI 0 Data Register
PORTS:       equ $D6      ;Port S Data Register
DDRS:        equ $D7      ;Port S Data Direction Register
Upper_Byte:  equ $0B00
Lower_Byte:  equ $0B01

************************************************
* Main Program
************************************************
ORG     $0800             ;User code data area, start main program at $0800
DATA    FCB  00,01        ;Set up 16 bit DATA variable format
MAIN:
        BSR INIT          ;Subroutine to initialize SPI registers
        BSR SAMPLE        ;Subroutine to start transmission
```

## BRA sample program code (Continued)

```
INIT:
       BSET DDRS, #%11101100      ;Configure PORT S inputs/outputs:
                                  ;SS/CS, SCK, MOSI, MISO, PS3, PS2, TXD,
                                  ;RXD
       BSET SP0BR, #%00000001     ;Set Baud Rate
       BSET SP0CR1, #%01010100    ;SPI Configuration Register 1(SP0CR1):
                                  ;SPIE, SPE, SWOM, MSTR, CPOL, CPHA, SSOE,
                                  ;LSBF
       BSET SP0CR2, #%00000000    ;SPI Configuration Register 2 (SP0CR2):
                                  ;-,-,-,-,-,-,SSWAI, SPCO
       MOVB #$00, UPPER_BYTE      ;Set upper byte to zero
       MOVB #$00, LOWER_BYTE      ;Set lower byte to zero
       RTS                        ;Return from initialization subroutine

*********************************************************************
* Sample and convert here
*********************************************************************
SAMPLE:
       MOVB #$08, PORTS           ;Sets ADC CS* LOW, DAC CS* HIGH, FS HIGH
       MOVB #$00, SP0DR           ;Write zero value to data register
                                  ;to generate SCLK for ADC
       BSR FLAG                   ;Clear SPIF
       LDAA SP0DR                 ;Load first ADC Sample (Upper Byte)
       MOVB #$00, SP0DR           ;Write zero value to data register
                                  ;to generate SCLK for ADC
       BSR FLAG                   ;Clear SPIF
       LDAB SP0DR                 ;Load second ADC Sample (Lower Byte)
       STD DATA                   ;Store ACCA and ACCB in Data

       LDX #$0004                 ;Load value 4 in X
LOOP:                             ;Start loop to shift right by four
       LSRD                       ;Right Shift ACCD
       DBNE X,LOOP                ;Loop "X" times
       STAA UPPER_BYTE            ;Store accumulator A in Upper Byte
       STAB LOWER_BYTE            ;Store accumulator B in Lower Byte

       MOVB #$88, PORTS           ;Set DAC CS* LOW, ADC CS* HIGH, FS HIGH
       BCLR PORTS, #%00001000     ;Set FS to DAC(PS3) LOW
       MOVB UPPER_BYTE, SP0DR     ;Load Data Register with Upper Byte
       BSR FLAG                   ;Clear SPIF
       MOVB LOWER_BYTE, SP0DR     ;Load Data Register with Upper Byte
       BSR FLAG                   ;Clear SPIF

       BSET PORTS, #%00001000     ;Set FS for the DAC HIGH
       BRA SAMPLE                 ;Go back and take another sample

FLAG:
       BRCLR SP0SR,#$80,FLAG      ;Wait for flag to clear.
       RTS
```

## References

For more information related to this article, you can download an Acrobat Reader file at www-s.ti.com/sc/techlit/*litnumber* and replace *"litnumber"* with the **TI Lit. #** for the materials listed below.

| Document Title | TI Lit. # |
|---|---|
| 1. TLV2541 Data Sheet | slas245 |
| 2. TLV5636 Data Sheet | slas223 |
| 3. TMS320C31 Data Sheet | sprs035 |
| 4. Motorola, MC68HC912B32 Technical Summary | — |

## Related Web sites

**www.dataconverter.com**

**dsp.ti.com/ccstudio**

**www.ti.com/sc/docs/products/analog/***device***.html**
Replace *device* with tlc2551, tlc2552, tlc2555, tlv2541, tlv2542, tlv2545, tlv5606, tlv5616, tlv5617a, tlv5618a, tlv5623, tlv5624, tlv5625, tlv5626, tlv5636, tlv5637, or tlv5638

**www.ti.com/sc/docs/products/dsp/tms320c31.html**

**email TI's Applications Group**
Send questions regarding this data converter to:

dataconvapps@list.ti.com

For a faster response, insert the device part number or EVM number in the subject heading.

## 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** | |
| 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** |

# *TI Worldwide Technical Support*

## *Internet*

**TI Semiconductor Product Information Center Home Page**
support.ti.com

**TI Semiconductor KnowledgeBase Home Page**
support.ti.com/sc/knowledgebase

## *Product Information Centers*

**Americas**

| | | | |
|---|---|---|---|
| Phone | +1(972) 644-5580 | Fax | +1(972) 927-6377 |
| Internet/Email | support.ti.com/sc/pic/americas.htm | | |

**Europe, Middle East, and Africa**

| | | | |
|---|---|---|---|
| Phone | | | |
| Belgium (English) | +32 (0) 27 45 54 32 | Netherlands (English) | +31 (0) 546 87 95 45 |
| Finland (English) | +358 (0) 9 25173948 | Russia | +7 (0) 95 7850415 |
| France | +33 (0) 1 30 70 11 64 | Spain | +34 902 35 40 28 |
| Germany | +49 (0) 8161 80 33 11 | Sweden (English) | +46 (0) 8587 555 22 |
| Israel (English) | 1800 949 0107 | United Kingdom | +44 (0) 1604 66 33 99 |
| Italy | 800 79 11 37 | | |
| Fax | +(49) (0) 8161 80 2045 | | |
| Internet | support.ti.com/sc/pic/euro.htm | | |

**Japan**

| | | | |
|---|---|---|---|
| Fax | | | |
| International | +81-3-3344-5317 | Domestic | 0120-81-0036 |
| Internet/Email | | | |
| International | support.ti.com/sc/pic/japan.htm | | |
| Domestic | www.tij.co.jp/pic | | |

**Asia**

| | | | |
|---|---|---|---|
| Phone | | | |
| International | +886-2-23786800 | | |
| Domestic | Toll-Free Number | | Toll-Free Number |
| Australia | 1-800-999-084 | New Zealand | 0800-446-934 |
| China | 800-820-8682 | Philippines | 1-800-765-7404 |
| Hong Kong | 800-96-5941 | Singapore | 800-886-1028 |
| Indonesia | 001-803-8861-1006 | Taiwan | 0800-006800 |
| Korea | 080-551-2804 | Thailand | 001-800-886-0010 |
| Malaysia | 1-800-80-3973 | | |
| Fax | 886-2-2378-6808 | Email | tiasia@ti.com |
| Internet | support.ti.com/sc/pic/asia.htm | | ti-china@ti.com |

**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