SPI-Based Data Acquisition/Monitor Using the TLC2551 Serial ADC

Tom Hendrick *Advanced Analog Applications*

ABSTRACT

The TLC2551 12-bit serial output analog to digital converter can easily interface to the serial peripheral interface port of many popular microcontrollers. This application report focuses on using the SPI port of the MSP430F149 and 68HC912 microcontroller in a complete data acquisition system using the TLC2551 analog to digital converter and the TLV5616 digital to analog converter. Project collateral discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/SLAA108.

Contents

Figures

Tables

For questions regarding this or other data converter products, e-mail the Data Converter Applications Team at dataconvapps@list.ti.com. Please include the product name in the subject heading.

Introduction

The TLC2551 ADC requires a falling clock edge while the chip select pin is in the high state in order to release the output data. This requirement can be satisfied by writing to a second device on the serial bus, while holding the ADC's chip select pin high. This application report describes how to use the TLC2551 ADC in conjunction with the TLV5616 DAC, creating a complete data acquisition system.

In this application report, an I/O pin on the microprocessor is assigned to perform the duties of frame sync for the digital to analog converter. The DAC's chip select pin is permanently held low. An alternative approach would be to use the inverted output of the SS/CS pin from the microcontroller as the DAC chip select.

The output from the TLV5616 can be used as the feedback signal in a control loop or as a system monitor.

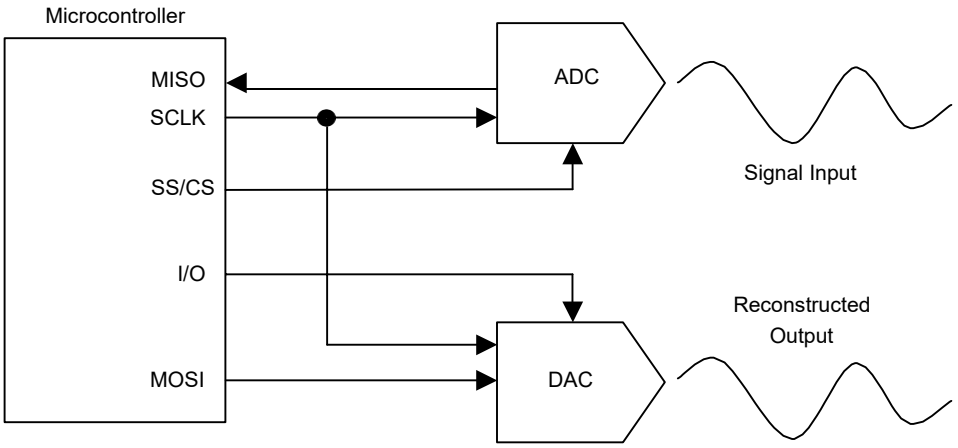

Figure 1. System Block Diagram

System Timing

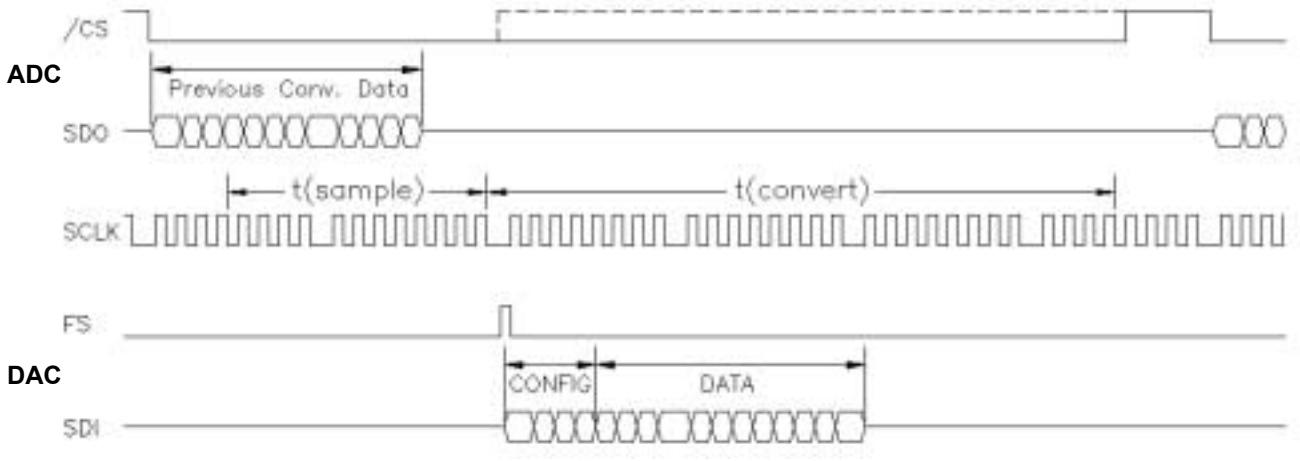

Figure 2. System Timing Diagram

A to D Requirements

The 12-bits of serial output data from the TLC2551 are presented MSB first after the falling edge of chip select. The ADC's chip select pin must be held low through two SPI transfers in order to allow the ADC to output the contents of its data register. Output data changes on the rising edge of SCLK, and is valid on the falling. The ADC also uses this time to complete the next data sampling cycle. For data to be released on the next chip select cycle, the ADC requires at least one falling SCLK edge while its chip select pin is in the high state. Asserting the micros SS/CS pin high before the completion of conversion, satisfies this requirement. Frame sync to the ADC is held high in this application.

D to A Requirements

Output data from the ADC is stored as two 8-bit words in the microprocessor. Before the data can be transmitted to the TLV5616 it must be formatted. Since the DAC is expecting the first data nibble to contain configuration information, the data received from the ADC must be padded with leading zeros by shifting the data right by four bits. Conversion speed and power down commands can be inserted by masking the shifted data.

The DAC's chip select is held low in this application. Toggling a general-purpose I/O pin on the micro can generate frame Sync to the DAC. This alerts the DAC to an incoming serial data stream. Data is presented to the DAC MSB first, in two subsequent SPI transfers from the microcontroller. The first four bits of data contain the configuration information, while the remaining 12-bits are data from the previous ADC conversion results.

TLC2551 Serial Interface

The TLC2551 is a 12-bit serial output successive approximation analog to digital converter. The device is available in 8-pin MSOP and SOIC packages. Since there are no configuration registers in the device, it is relatively simple to use. Table 1 lists the I/O pins and their function.

Table 1. I/O Pins of the TLC2551

Pin	Description
/CS	Chip select – An active low signal. A high-to-low transition removes the SDO pin from 3-state.
V_{REF}	External reference input
GND	Ground return for internal circuitry
AIN	Analog input
SCLK	Input serial clock
V_{DD}	Positive supply voltage (5VDC)
FS	Frame sync
SDO	3-state serial output for the conversion result.

TLC2551 Timing

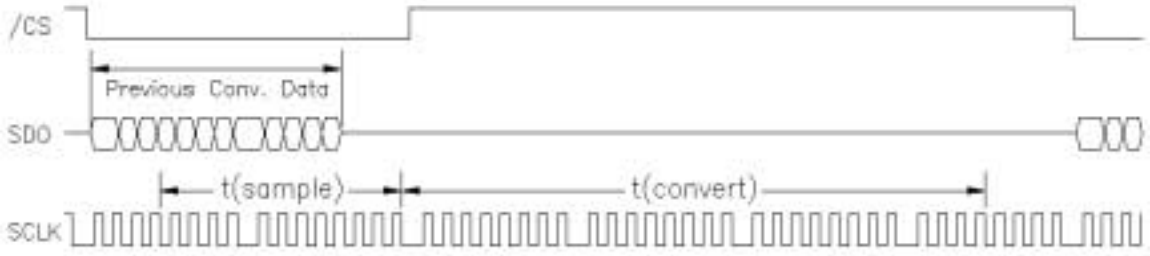

Figure 3. TLC2551 Timing Diagram

The TLC2551 requires 12 SCLK's to complete its sampling. The sampling period begins after the fourth falling SCLK edge. Chip select must be held in a low state for the entire 16 clock cycles before a valid conversion will take place. When operated with an SPI controller, the chip select signal can be removed after two 8-bit transfers from the SPI port.

The conversion cycles starts after the 16th falling SCLK edge and takes 28 clock cycles to complete. The results are presented on SDO during the next cycle.

TLV5616 Serial Interface

The TLV5616 is a 12-Bit voltage output digital-to-analog converter (DAC) with a serial interface. When the device is enabled - /CS held low – a falling edge of frame sync starts shifting data to the internal register. The input data format is a 16-bit serial string which contains four control and 12 data bits.

Table 2. I/O Pins of the TLV5616

Pin	Description
DIN	Serial input data – 16-Bit format, first 4 MSB's are control bits, followed by 12 data bits
SCLK	Serial input clock
/CS	Chip select – An active low signal used to enable and disable inputs
FS	Frame sync – This pin is used in DSP based systems to synchronize data transfers. In microcontroller systems, this pin can be used as chip select
AGND	Analog ground
REFIN	Reference analog voltage
OUT	DAC analog output voltage
V_{DD}	Analog supply voltage (2.7-5.5VDC)

The TLV5616 uses a 16-bit data format with the first four MSB's containing device programming information, followed by the 12-bits of serial data to be converted. Data bit D14 controls the conversion speed, while D13 provides a software power down. Setting D14 puts the device in fast conversion (3uS) mode. Setting D13 puts the TLV5616 in power down (10nA) mode.

	15	14	13	12	11 - 0
Din	X	SPD	PWR	X	Data for Conversion

Figure 4. TLV5616 Data Register

TLV5616 Timing

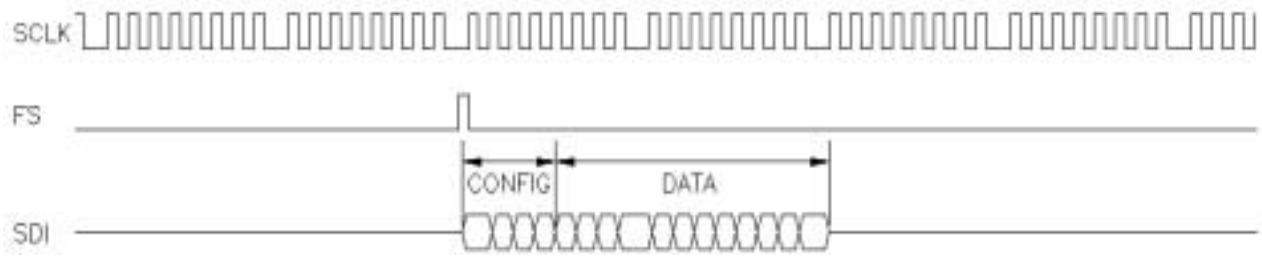

Figure 5. TLV5616 Timing Diagram

Note: The DAC's chip select pin is held low in this application and therefore is not shown on the timing diagram.

MSP430F149 Code Example

```
;*******************************************************************
; MSP430F149 Demo - Data Acquisition with the TLC2551 ADC and TLV5616 DAC
;
;
;               TLC2551          MSP430F149
;            -------------       -------------
;          |              |    |             |
;          |           SDO|--->|P3.2(SOMI)   |
;   CH0~~>|AIN       SCLK|<---|P3.3(UCLK)   |
;          |           /CS|<---|P3.0(GPIO)   |
;          |            FS|<---|P3.5(GPIO)   |
;          |              |    |             |
;          |              |    |             |
;                              |             |
;                              |             |
;             TLV5616 DAC      |             |
;             _____________    |             |
;          |           SDI|<---|P3.1(SIMO)   |
;          |           /CS|<---|P5.3(GPIO)   |
;          |            FS|<---|P3.5(GPIO)   |
;
; Note:  MSP430F149 is set with 8Mz Resonator in this code.
;
; Assembled with IAR Embedded Workshop for MSP430 Kickstart
;
; Texas Instruments, Inc.
; Tom Hendrick
; Data Aquisition Applications - Dallas
; Nov. 2001
;*******************************************************************
#include   "msp430x14x.h"          // Standard Equations

#define DUMMY               0x000

;*******************************************************************
;Constants
;*******************************************************************
```

```
ADC_CS      equ    001h                  ; Assign p3.1 to ADC_CS
FS          equ    010h                  ; Assign p3.6 to FS (DAC and ADC)
DAC_CS      equ    020h                  ; Assign p5.4 to DAC_CS

;-------------------------------------------------------------------------
;Setup Stack
;-------------------------------------------------------------------------
            RSEG  CSTACK
            DS    0

;-------------------------------------------------------------------------
;Program Code
;-------------------------------------------------------------------------
            RSEG CODE
;-------------------------------------------------------------------------
RESET_ISR   mov     #SFE(CSTACK),SP     ; define stackpointer
            call    #Init_Sys           ; Initialize the MSP430
            call    #Mainloop           ; Run the Main Program
Mainloop
Read_ADC    bic.b #ADC_CS,&P3OUT
            mov.b #DUMMY,&U0TXBUF       ; Dummy write to SPI (generates SCLK)
            call #CLEAR
            mov.b &U0RXBUF,R12          ; Store Upper Byte
            mov.b #DUMMY,&U0TXBUF       ; Dummy write to SPI (generates SCLK)
            call     #CLEAR
            mov.b &U0RXBUF,R13          ; Store Lower Byte

            mov.b #04h, R7
CLK_LOOP    mov.b #DUMMY,&U0TXBUF       ; Dummy write to SPI
CONV_CLK    bit.b #01h,&IFG2            ; TXBUF ready?
            jnc CONV_CLK                ; 1 = ready
            bic.b #01h,&IFG2
            dec R7
            cmp #00h, R7
            jnz CLK_LOOP
            biC.b #ADC_CS,&P3OUT

;           jmp Mainloop                ; Repeat
; Remove remark from above line to skip transmit back to DAC
```

```
Write_DAC  bis.b #001h,&P3OUT         ; Turn on LED of F149 Board
           swpb     R12                   ; Swap Bytes
           add.w R12, R13
           and.w #0xFFF0, R12         ; Strip trailing bits
           rrc.w    R12           ; Shift data 4 places
           rrc.w    R12           ;     to conform to
           rrc.w    R12           ; DAC input format
           rrc.w    R12           ; Data is shifted!
           and.w #0x0FFF, R12         ; Strip any carries
           add.w #0x4000, R12         ; Set DAC Fast Mode, 0x4000

           bic.b #FS,&P3OUT           ; Clear FS to DAC
           bic.b #DAC_CS,&P3OUT       ; Enable /CS to DAC
           bis.b #FS,&P3OUT           ; Toggle FS to prepare DAC
           bic.b #FS,&P3OUT           ; to recieve DATA
           swpb R12               ; Align MSB First
           mov.b R12,&U0TXBUF         ; Transmit upper Data Byte to DAC
           call #CLEAR
           swpb R12                   ; Prepare Lower Byte
           mov.b R12,&U0TXBUF         ; Transmit lower Data Byte to DAC
           call #CLEAR
           bis.b #DAC_CS,&P3OUT       ; Disable /CS to DAC
           bis.b #FS,&P3OUT           ; Set Frame Sync
           bic.b #001h,&P3OUT         ; Turn off LED of F149 Board
           jmp Mainloop               ; Repeat

;******************************
; Clear TX Flag
;******************************
CLEAR
           bit.b # UTXIFG0,&IFG1; TXBUF ready?
           jnc CLEAR                  ; 1 = ready
           bic.b # UTXIFG0,&IFG1
           ret
;-------------------------------------------------------------------------
Init_Sys;  Modules and Controls Registers set-up subroutine
;-------------------------------------------------------------------------
StopWDT    mov     #WDTPW+WDTHOLD,&WDTCTL  ; Stop Watchdog Timer
```

```
SetupClock
          bic.b #XTOFF, &BCSCTL1
          bis.b #SELM2+SELS, &BCSCTL2
HF_WAIT  ; 8MHz Crystal used - wait for stabilization
          bic.b #OFIFG, &IFG1
          bit.b #OFIFG, &IFG1
          jnz HF_WAIT
          bic.b #OFIFG, &IFG1             ; Clear Oscillator fault flag
          bit.b #OFIFG, &IFG1             ; Test for clear
          bis.b #001h,&P1DIR               ; P1 pin 1 set to output (toggle's LED)

Setup_SPI0_Port
          bis.b #00Eh,&P3SEL               ; P3.1,2,3 SPI option select
          bis.b #ADC_CS+FS+DAC_CS,&P3DIR  ; /CS, FS = P3 output direction
          bis.b #ADC_CS+FS+DAC_CS,&P3OUT  ; P3.5,6,7 CS & FS set

SetupInterrupt
          bic.b #02h, &P1IFG              ; Clear interrupt flags

SetupSPI0
          bis.b #040h,&ME1             ; Enable SPI TX/RX
          mov.b  #CHAR+SYNC+MM,&U0CTL ;  8-bit  SPI  Master
          bis.b #SSEL0+SSEL1+STC,&U0TCTL
          mov.b #02h,&U0BR0            ; Set SPI Baud Rate
          mov.b #00h,&U0BR1            ; This give 4MHz SCLK w/ 8MHz Crystal
          mov.b #00h,&U0MCTL
          ret

;-------------------------------------------------------------------------------
         COMMON  INTVEC              ; MSP430x11x1/MSP430F14x Interrupt vectors
;-------------------------------------------------------------------------------
          ORG     RESET_VECTOR
RESET_VEC  DW      RESET_ISR           ; POR, ext. Reset, Watchdog
END
```

68HC912 Code Example

```
;*******************************************************************************
;       Motorola 68HC912 Demo - SPI Communication with TLC2551 and TLV5616
;       Using the M68EVB912B32 Demo Bd. with DeBug12
;       Assembled with ASM12.EXE
;            TLC2551           68HC912           TLV5616
;        -------------     -------------     --------------
;       |          CS|<---|PS.7    PS.3|--->|FS            |
;  ~~~~>|IN+     SCLK|<---|    PS.6    |--->|SCLK     OUT+|~~~~>
;       |         SDO|--->|PS.4    PS.5|--->|SDI           |
;       |            |    |            |    |              |
;
;       T. Hendrick
;       Analog Applications Group
;       Texas Instruments, Inc
;       August 2000
;*******************************************************************************
;* -----------------------------------------------------------------------
;* Equates and Variables
;* -----------------------------------------------------------------------
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
; User Variables
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
MAIN:
```

```
     BSR     INIT                ; Subroutine to initialize SPI registers
     BSR     SAMPLE              ; Subroutine to start transmission

;* ---------------------------------------------------------------------
;*           Initialization Subroutine
;* ---------------------------------------------------------------------
INIT:
     BSET DDRS, #%11101100       ; Configure PORT S input/ouput:
                                 ; SS/CS, SCK, MOSI, MISO, PS3, PS2, TXD, RXD
     BSET SP0BR, #%00000000      ; Set Baud Rate
     BSET SP0CR1, #%01010100     ; Configure SPI(SP0CR1):
                                 ; SPIE, SPE, SWOM, MSTR, CPOL, CPHA, SSOE, LSBF
     BSET SP0CR2, #%00000000     ; Configure SPI(SP0CR2):
                                 ; -,-,-,-,-,-,SSWAI, SPCO
     MOVB #$00, UPPER_BYTE
     MOVB #$00, LOWER_BYTE
     RTS

;* ----------------------------------------
;*     Sample / Convert
;* ----------------------------------------
SAMPLE:
     MOVB #$00, PORTS            ; Sets ADC CS Lo
     MOVB #$00, SP0DR            ; Write zero value to data register
                                 ; to generate SCLK for ADC
     BSR FLAG                    ; Clear SPIF
     LDAA SP0DR                  ; Load first ADC Sample
     MOVB #$00, SP0DR            ; Write zero value to data register
                                 ; to generate SCLK for ADC
     BSR FLAG                    ; Clear SPIF
     LDAB SP0DR                  ; Load second ADC Sample
     STD DATA                    ; Store ACCA and ACCB in Data

;* ---------------------------------------------------------------------
;* Data Shifting Subroutine
;* ---------------------------------------------------------------------
     LDX #$0004                  ; Shift data four bits to
                                 ; bypass DAC config registers
```

```
LOOP:
     LSRD                        ; R Shift ACCD
     DBNE X,LOOP                 ; Loop 4 times
     STAA UPPER_BYTE             ; Store shifted data
     STAB LOWER_BYTE             ; Store shifted data

;* ---------------------------------------------------------------------
;* Transmit Subroutine
;* ---------------------------------------------------------------------
     MOVB #$80, PORTS            ; ADC CS Hi
     BSET PORTS, #%00001000      ; Set FS to DAC(PS3) Hi
     BCLR PORTS, #%00001000      ; Set FS to DAC(PS3) Lo
     MOVB UPPER_BYTE, SP0DR      ; Load Data Register with Upper Byte
     BSR FLAG
     MOVB LOWER_BYTE, SP0DR      ; Load Data Register with Upper Byte
     BSR FLAG
     MOVB #$00, SP0DR            ; Load Data Register with Zero
     BSR FLAG
     MOVB #$00, SP0DR            ; Load Data Register with Zero
     BSR FLAG
     BRA SAMPLE                  ; Go back and do it again

;* ----------------------------------------
;* Clear SPIF
;* ----------------------------------------
FLAG:
     BRCLR     SP0SR,#$80,FLAG      ; Wait for flag.
     RTS
```

References

1. *Multi Converter EVM Users Guide* (SPRU131)
2. *MSP430x1xx Family User's Guide* (SLAU049)
3. *Motorola Semiconductor Technical Data – MC68HC912B32* (MC68HC912B32TS/D)
4. *TLV5616 data sheet* (SLAS152)
5. *TLC2551 data sheet* (SLAS276)

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

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