ADC Interfacing with pt-51 using SPI

26 September 2019

Objectives:

1. Learn embedded C
2. Understanding the Serial Peripheral Interface (SPI) protocol
3. Write driver code for interfacing ADC IC TLV1543 with Pt-51 using SPI
4. Integrating ADC and LCD driver programs to display an analog value using Pt-51

Introduction

SPI (Serial Peripheral Interface) is a synchronous, serial communication protocol used for communicating between microcontrollers and peripherals (or between 2 or more microcontrollers). In SPI communication, there is a master device (typically the microcontroller), and a slave device (typically a peripheral or other microcontroller). It uses shift registers for converting parallel data to serial and four lines for communication. They are two data lines viz MOSI (Master Out Slave In), MISO (Master In Slave Out), a CLK, and a Select line to choose the device you wish to communicate to.

In this experiment, we will use TLV1543, a 11 channel, 10-bit switched-capacitor, successive-approximation ADC. The analog signal to be applied may be single ended or differential. Any of the 11 analog input channels can be selected by sending address of corresponding channel serially to the address pin of ADC. The ADC produces 10 bit digital output which is available serially on the Dout pin of the ADC. Please refer to the datasheet of TLV1543 for more details.

Please note that this experiment on-wards we shall use embedded C for programming.

Homework

(Read the following documents in this order)

1. Read the supporting document 'SPI_intro.pdf'
2. Read Serial Peripheral Interface (page 89) part of AT89C5131 datasheet, understand the timing diagram, SPCON register, SPSTA register

3. Read TLV1543 datasheet, give special attention to timing diagrams, modes of operation of TLV, allowable max frequency
4. Read this handout.
5. Read the document on "Embedded C".
6. Go through all shared codes.

Circuit diagram

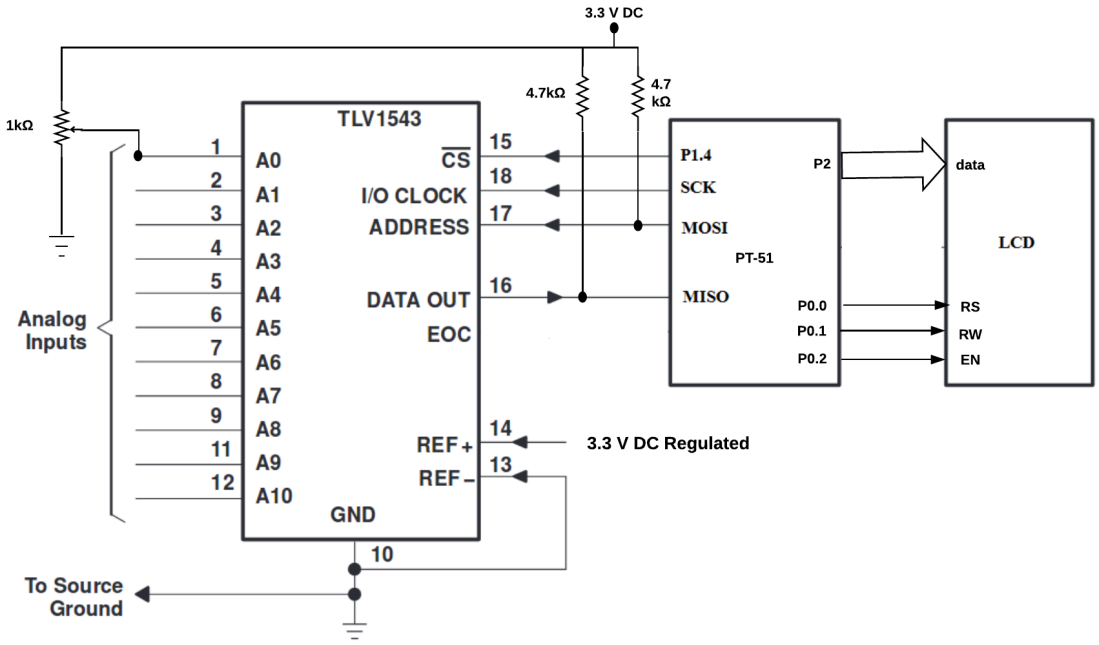


Figure 1: Circuit Diagram

Pt-51 Board			TLV 1543 ADC		
Pin	**Function**	**Description**	**Pin**	**Function**	**Description**
P1.7	MOSI	Serial Output	17	Address	Serial Input
P1.6	SCK	Serial Clock O/p	18	I/O clk	Serial Clock I/P
P1.5	MISO	Serial Input	16	Data out	Serial O/p
P1.4	$\overline{SS}$	Slave Select	15	$\overline{CS}$	Chip Select

Table 1: SPI pin connections between Pt-51 and TLV1543

The pins on the Pt-51 board marked as SPI are used to connect peripherals with SPI connectivity, i.e., the ADC TLV 1543 in this experiment. Understand the circuit in Fig. 1 and connections in Table 1 for the required hardware connection.

Software architecture

The program flow and function calls can be understood from Fig. 2.

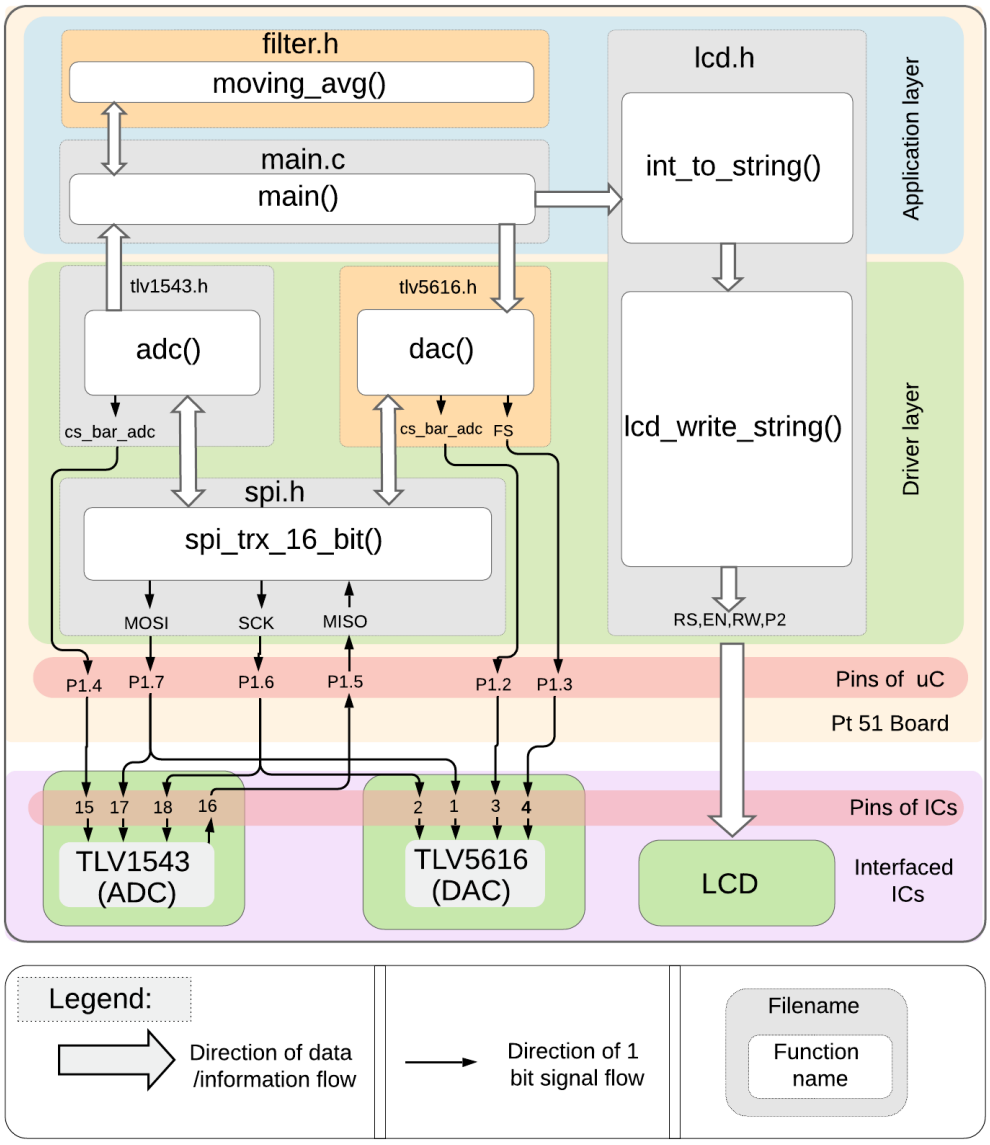


Figure 2: Software architecture for interfacing the SPI peripherals. You can ignore orange coloured blocks (filter.h and TLV5616.h) for now as we will use them in the next experiment.

In this experiment, we will be performing several tasks and hence we shall follow a modular coding practice (writing header files for the tasks and functions (subroutines) for each subtask). This practice makes it easier to read, understand and debug the code. It also makes it possible to scale up the project by including additional header files. (In fact, we will be adding another module to this experiment during the next lab session.)

Function descriptions

This section describes all the functions required to be written for the interfacing of ADC to read analog DC input voltage connected to one of its channels and display it on the LCD. Note that each header file and functions in them perform specific tasks. The functions are written such that they take input data from other function and return processed data.

1. **main.c :**
   This file includes all necessary header files which have function definitions that will be used. The microcontroller executes *main()* function from this file. This function calls all necessary functions sequentially. Use *main.c* as it is, there is no need to edit this file. Flow of operations in *main()* is as follows:

   (a) It calls initialization functions for all peripherals which are *spi_init()* (for SPI module of Microcontroller AT89C5131), *adc_init()* (for ADC IC TLV1543) and *lcd_init()* (for LCD).

   (b) Display "Channel-00" on first line using functions available in **lcd.h**.

   (c) Steps (e) to (g) are repeated infinitely using while loop so that recent input sample will be displayed on LCD.

   (d) Call *adc()* function for communicating with the ADC, which will return digital values in the range 0 to 1023. This value is proportional to the input voltage. Note that input voltage to be applied is 0 to 3.3 V.

   (e) Appropriately scale the digital value (0 to 1023) to lie in the 0 to 3300 mV range.

   (f) Call *int_to_string()* function with appropriate argument to convert sampled input value to an ascii string that can be displayed on the LCD.

   (g) Call function *lcd_write_string()* to display the ascii value on the 2nd line of LCD.

2. **spi.h :**
   Functions in this file control all operations related to AT89C5131 SPI module. Task of this module is to facilitate SPI communication between master (Microcontroller) and slave (SPI device - ADC). Data from all interfaced SPI devices are gathered by calling function from this file and passed to other functions for further processing. So if we have to switch to another microcontroller plateform, only **spi.h** (driver) file needs to be changed and processing functions can remain the same. The file **spi.h** contains following functions.

   (a) *spi_init():*

      i. Parameters to be passed: None

      ii. Parameters to be returned: None

      iii. Task:
      Function initialises SPI module in AT89C5131 by configuring SPCON, IEN0 and IEN1 register.

   (b) *spi_trx_16_bit()*:

i. Parameters to be passed: unsigned int - 16 bit data to be sent from master to slave.

ii. Parameters to be returned: unsigned int - 16 bit data from slave.

iii. Task:
In AT89C5131, spdat (Transmit and receive data buffer) is an 8 bit SFR which allows 8 bit data to be transmitted or received. To transmit or receive 16 bit data, two 8 bit data frames may be used one after the other. 16 bit data passed to this function from other functions will be unpacked (split) into two bytes. These two bytes are transmitted to slave, simultaneously master will receive two bytes from the slave. Note that most significant byte is transmitted first and then the least significant byte. Similarly, the master will receive the most significant byte first and then the least significant byte. These two received bytes are then packed (joined) into 16 bit data. This data word will be returned by this function.

(c) *spi_interrupt()*:

i. Parameters to be passed: None

ii. Parameters to be returned: None

iii. Task: This is the ISR (Interrupt Service Routine) for SPI interrupt. This function will be called after completion of transfer of 8 bits of data to the slave. The 8 bits received from the slave are stored in a temporary global variable which will be used by *spi_trx_16_bit()* function.

3. **tlv1543.h:**
Functions in this file are specific to TLV1543. These functions control and enable data transfer between microcontroller and ADC slave. Handling of control signals and data processing specific to this IC is facilitated by this header file. Functions from **spi.h** are called by functions in this file. Hence **spi.h** is included in *tlv1543.h*. Configuration parameters specific to TLV1543 are set in function in this file. Following are functions in this file.

(a) *adc_init()*:

i. Parameters to be passed: None

ii. Parameters to be returned: None

iii. Task:
Control signals related to TLV1543 are initialized here.

(b) *adc()*:

i. Parameters to be passed: None

ii. Parameters to be returned: unsigned int - two bytes of data corresponding to the analog input received.

iii. Task:
This function triggers communication between microcontroller and ADC slave. Following tasks are executed sequentially.

1) Multiple slaves (if connected) might need different settings of configuration of same SPI module of master. These slave specific requirements are taken care of by functions in slave specific driver (Here *adc()*).
2) Enable ADC slave.
3) A two byte data frame having address of input channel A0 is formed (Check timing diagram of SPI communication for this). This data is then passed to function *spi_trx_16_bit()*.
4) *spi_trx_16_bit()* will take care of SPI communication between microcontroller and ADC slave and return sampled ADC value from that channel.
5) Disable ADC slave.
6) Data word returned by function *spi_trx_16_bit()* is processed to make it in correct format (Check timing diagram of SPI communication for this). This data word is then returned by this function.

4. **lcd.h:**
You have already used LCD in previous experiments using assembly code. In this experiment, we shall write C functions for doing the same. **lcd.h** header includes all functions related to LCD module. Following are some key functions in **lcd.h** (other functions are self explanatory).

(a) *lcd_init()*:
   i. Parameters to be passed: None
   ii. Parameters to be returned: None
   iii. Task:
   This function initializes LCD. It clears LCD and move cursor to start of 1st line of LCD.

(b) *lcd_cmd()*:
   i. Parameters to be passed: unsigned int - 1 byte command for LCD.
   ii. Parameters to be returned: None
   iii. Task:
   This function activates control signal specific to sending command to LCD. It transfers 1 byte command to LCD module. Clear screen, move courser to specific location etc., are few commands. See datasheet of LCD for more commands.

(c) *lcd_write_char()*:
   i. Parameters to be passed: unsigned char - 1 byte, ASCII value of the character.
   ii. Parameters to be returned: None
   iii. Task:
   Activate control signal specific to printing a character on LCD. This function transmits ASCII value of the character to LCD. LCD prints it at current position of cursor. Cursor will be moved to next position.

(d) *int_to_string(,)*:

i. Parameters to be passed: unsigned int - 16 bits of positive integer data and unsigned char * - pointer to an ASCII string location.

ii. Parameters to be returned: None

iii. Task:
This function converts positive integer value to decimal number. This decimal number is then converted to string of ASCII. This string is stored at location pointed by the pointer.

(e) *lcd_write_string()*:

i. Parameters to be passed: unsigned char * - Pointer to an ASCII string which is to be displayed on the LCD.

ii. Parameters to be returned: None

iii. Task:
This function uses function *lcd_write_char()* to send all characters from ASCII string to LCD.

Labwork

1. Complete function *spi_init()* to configure SPCON, IENO and IEN1 registers of AT89C5131.

   (a) microcontroller SPI module as Master

   (b) Free the SS pin for a general-purpose (microcontroller does not requires chip select as we are configuring it in master mode)

   (c) Select the Master clock rate as Fclk/16 (by choosing appropriate values of SPR0, SPR1, SPR2 we can get different baud rates)

   (d) Select appropriate setting for serial clock polarity and clock phase: Refer timing diagrams (Figures 3 and 4 in this handout) of SPI communication of microcontroller and TLV1543 (Refer datasheets of both ICs for more information).

   (e) Enable SPI module of the microcontroller.

   (f) Enable SPI interrupt (**IEN1**-Interrupt enable register) (Read the datasheet of AT89C5131a for more information)

   (g) Set enable all interrupt (EA) bit (**IEN0**- Interrupt enable register) (Refer datasheet of AT89C5131a for more information)

2. Connect circuit as mentioned in Fig. 1. Note that input is given to A0 channel. Display the measured voltage (0 to 3.3 V) on the LCD in the format: On 1st line "Channel-00:" and on 2nd line "XXXXX mV" (XXXXX is sampled input voltage from channel A0 in mV).

3. Modify function *adc()* such that it will take 1 byte data as an input argument. This argument takes value from 0 to 13 which corresponds to input channel from A0 to A10 and 3 internal test inputs of ADC (Refer TLV1543 datasheet page no 5 and 6). Function *adc()* should return sampled input corresponds to ADC channel number

which was passed to it. Update *main()* function such that input from any two channels (A0 - A10) of ADC and three internal test inputs will be displayed periodically with an interval of 2 secs on LCD in following format: On 1st line "Channel-XX:" (XX - channel number: 0 to 13) and on 2nd line "XXXX mV" (XXXXX is sampled input voltage in mV from ADC channel number XX).

Formula for voltage conversion

Approximate procedure for analog to digital conversion of input voltage is given as (Refer figure number-7 for ideal conversion characteristics)

$$LSB\,Size = \frac{V_{REF}}{1023}$$

$$Digital\,Output\,Code = \frac{1023 \times V_{IN}}{V_{REF}}$$

Where,
V_{IN} = Analog input voltage
V_{REF} = Analog reference voltage

Details from datasheets of AT89c5131 and TLV1543

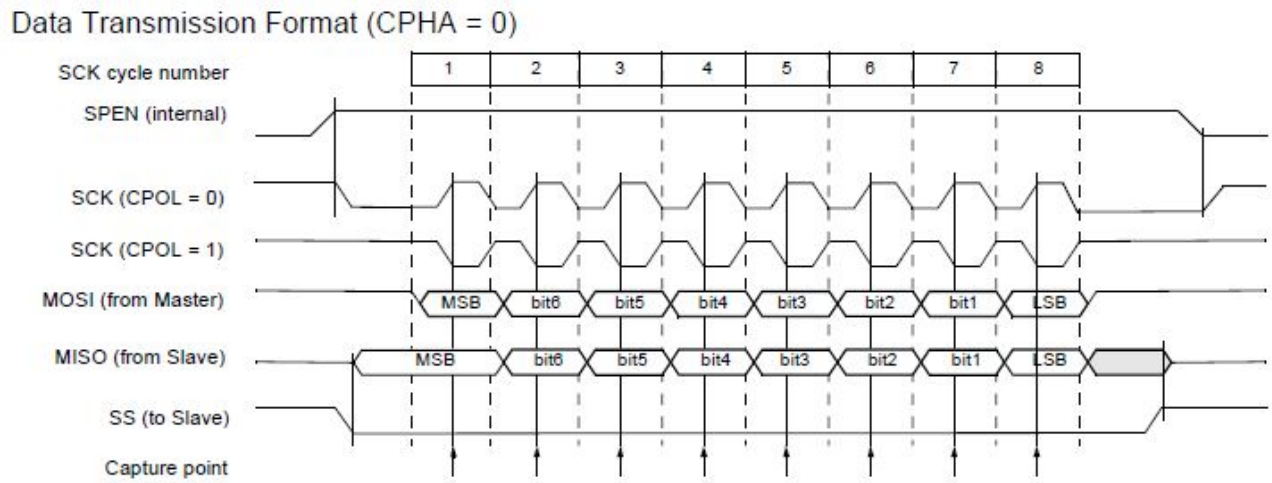
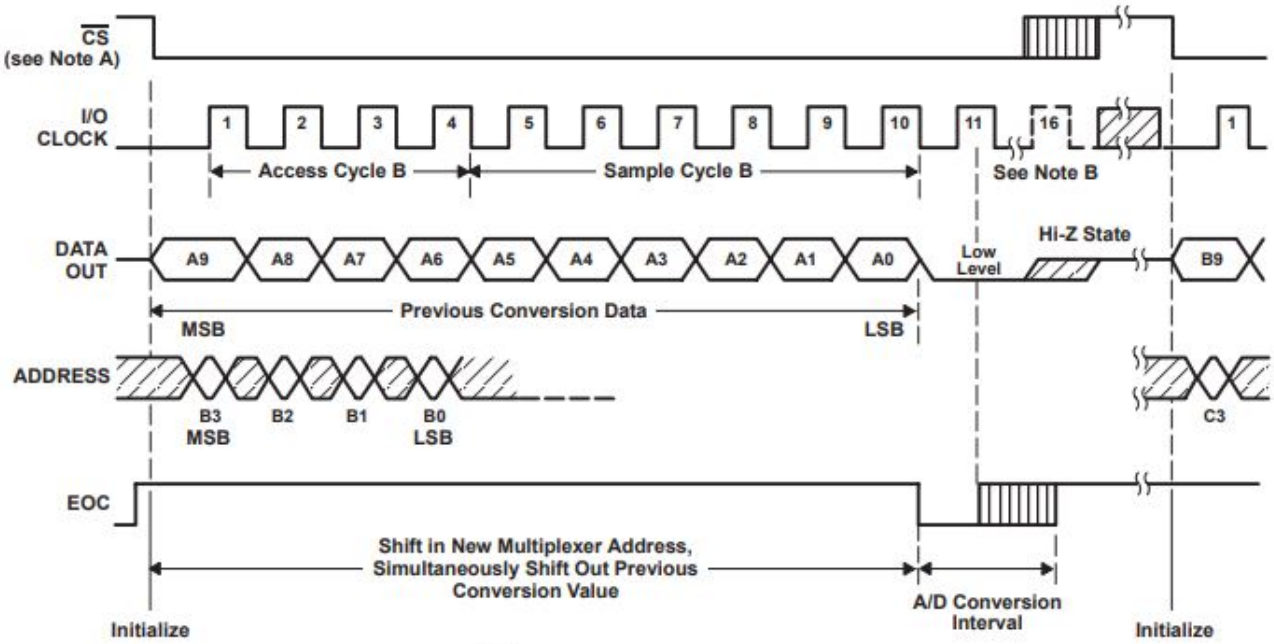


Figure 3: SPI timing diagram for AT89C5131



NOTES: A. To minimize errors caused by noise at $\overline{CS}$, the internal circuitry waits for a set up time plus two falling edges of the internal system clock after $\overline{CS}\downarrow$ before responding to control input signals. No attempt should be made to clock in an address until the minimum chip $\overline{CS}$ setup time has elapsed.

B. The eleventh rising edge of the I/O CLOCK sequence must occur before the conversion is complete to prevent losing serial interface synchronization.

Figure 13. Timing for 11- to 16-Clock Transfer Using $\overline{CS}$ (Serial Transfer Interval Longer Than Conversion)

Figure 4: SPI timing diagram for TLV1543 IC

SPCON Register

SPCON - Serial Peripheral Control Register (0C3H)

7	6	5	4	3	2	1	0
SPR2	**SPEN**	**SSDIS**	**MSTR**	**CPOL**	**CPHA**	**SPR1**	**SPR0**

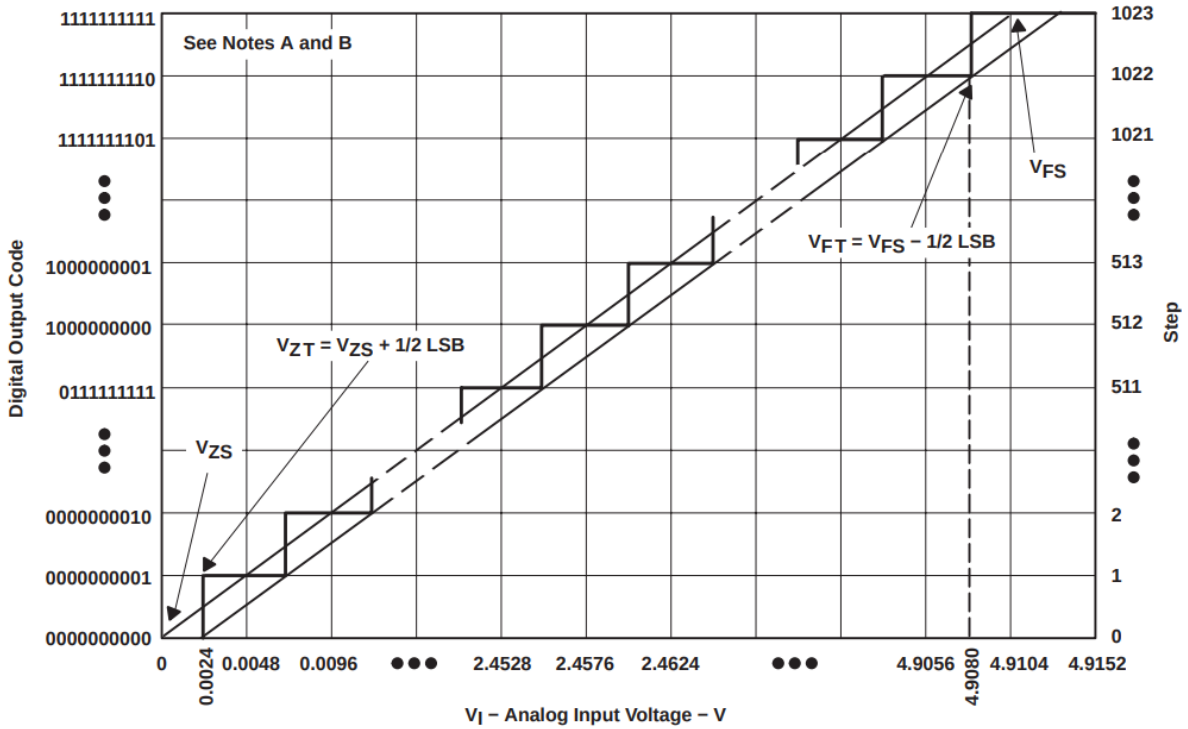
Bit Number	Bit Mnemonic	Description
7	SPR2	**Serial Peripheral Rate 2** Bit with SPR1 and SPR0 define the clock rate.
6	SPEN	**Serial Peripheral Enable** Cleared to disable the SPI interface. Set to enable the SPI interface.
5	SSDIS	**$\overline{SS}$ Disable** Cleared to enable $\overline{SS}$ in both Master and Slave modes. Set to disable $\overline{SS}$ in both Master and Slave modes. In Slave mode, this bit has no effect if CPHA = "0".
5	MSTR	**Serial Peripheral Master** Cleared to configure the SPI as a Slave. Set to configure the SPI as a Master.
4	CPOL	**Clock Polarity** Cleared to have the SCK set to "0" in idle state. Set to have the SCK set to "1" in idle low.
3	CPHA	**Clock Phase** Cleared to have the data sampled when the SCK leaves the idle state (see CPOL). Set to have the data sampled when the SCK returns to idle state (see CPOL).
2	SPR1	**SPR2 SPR1 SPR0 Serial Peripheral Rate** 0 0 0 Invalid 0 0 1 $F_{CLK\ PERIPH}/4$ 0 1 0 $F_{CLK\ PERIPH}/8$
1	SPR0	0 1 1 $F_{CLK\ PERIPH}/16$ 1 0 0 $F_{CLK\ PERIPH}/32$ 1 0 1 $F_{CLK\ PERIPH}/64$ 1 1 0 $F_{CLK\ PERIPH}/128$ 1 1 1 Invalid

Reset Value = 0001 0100b

Figure 5: SPCON Register: AT89C5131

1. The SPI module should be configured as a Master before it is enabled (SPEN set). Also the Master SPI should be configured before the Slave SPI.
2. The SPI module should be configured as a Slave before it is enabled (SPEN set).
3. The maximum frequency of the SCK for an SPI configured as a Slave is the bus clock speed.
4. Before writing to the CPOL and CPHA bits, the SPI should be disabled (SPEN ='0').

Figure 6: Note regarding SPI SFR, AT89C5131



NOTES: A. This curve is based on the assumption that V_{ref+} and V_{ref-} have been adjusted so that the voltage at the transition from digital 0 to 1 (V_{ZT}) is 0.0024 V and the transition to full scale (V_{FT}) is 4.908 V. 1 LSB = 4.8 mV.

B. The full-scale value (V_{FS}) is the step whose nominal midstep value has the highest absolute value. The zero-scale value (V_{ZS}) is the step whose nominal midstep value equals zero.

Figure 15. Ideal Conversion Characteristics

Figure 7: TLV1543 Analog to digital conversion