EE 337: DSP using SPI ADC-DAC Interface to Pt-51
Lab 8

Objectives:

1. Write driver code for interfacing DAC (TLV5616) with Pt-51.
2. Testing DAC (TLV5616) for a simple application for waveform generation.
3. Integrating DAC (TLV5616) and ADC (TLV1543) for implementing a moving average filter.

Background

We have used SPI (Serial Peripheral Interface) for interfacing ADC TLV1543 to Pt-51. In this week's experiment we shall use SPI to connect TLV5616 to Pt-51. The TLV5616 is a 12-bit DAC with 4-wire serial peripheral interface(SPI). The TLV5616 is programmed with a 16-bit serial string containing 4 control and 12 data bits.

In this experiment, we shall interface the DAC with Pt-51 to generate basic waveforms (ramp and sinusoidal) and observe them on DSO. We shall then use DAC, ADC and Pt-51 for implementing moving average filter.

Details regarding the DAC interfacing

The TLV5616 uses 4 wire lines for serial interfacing. The SCLK is used for clocking the chip and synchronizing it. DIN is the pin through which the 4 bit control signals along with the 12 bit data signal is received serially. The Frame Sync signal, FS, should be issued as per the timing diagram requirement to get the chip running. The chip select signal is used to select the chip when we connect and communicate with multiple chips using SPI protocol.

Note that here the data sampling happens at the falling edge of the SCLK signal. Therefore while making the configurations, also make the necessary changes to meet this requirement.

In the power down mode, all the amplifiers within the TLV5616 are disabled.

We would like to thank Texas Instruments for providing the DAC (TLV5616) for this experiment!

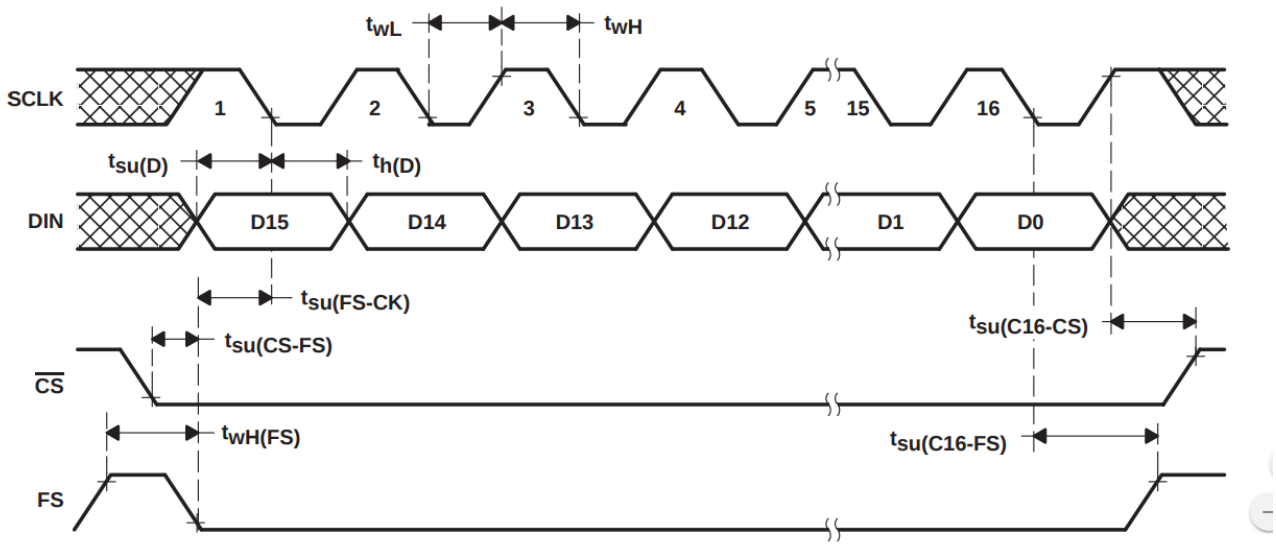

Figure 1: Timing diagram of TLV5616.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	SPD	PWR	X	New DAC value (12 bits)											

X: don't care
SPD: Speed control bit. 1 → fast mode 0 → slow mode
PWR: Power control bit. 1 → power down 0 → normal operation

Figure 2: Data format.

The output voltage (full scale determined by external reference) is given by:

$$Vout = 2 * V_{\text{ref}} * \frac{V_{digital\_in}}{2^n} \tag{1}$$

where V_{ref} is the reference voltage and $V_{digital\_in}$ is the digital input value within the range of 0 to $2^{(n-1)}$, where n = 12 (bits).

Homework

1. Read the datasheet of TLV 5616 for details from `http://www.ti.com/product/TLV5616/technicaldocuments`

2. Calculate the theoretical analog output values that can be obtained from the DAC for the following digital inputs($V_{ref} = \frac{3.3}{2}V$ and $V_{CC} = 3.3V$):

Digital input	**Expected output voltage**
0x000	
0x400	
0x800	
0xFFF	

3. Calculate the theoretical digital input values for which DAC give the following analog outputs($V_{ref} = \frac{3.3}{2}V$ and $V_{CC} = 3.3V$):

Analog output	**Required digital input**
1V	
1.66V	
3.3V	

Lab Work

1. **Interfacing TLV5616 to Pt-51**

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

      i. Microcontroller SPI module as Master

      ii. Free the SS pin (microcontroller does not require chip select as we are configuring it in master mode)

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

      iv. 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 TLV5616 (Refer datasheets of both ICs for more information).

      v. Enable SPI module of the microcontroller.

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

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

      Hint: Compare the timing diagrams of ADC (TLV1543) and DAC (TLV5616), you will see that there is a difference between the two. Note that the data is read by the peripheral in falling SCK edge for one and in the rising SCK edge for the other. SPCON values have to be set with this in mind.

   (b) Complete the function *dac()* in tlv5616.h. Refer the timing diagram to understand:

      i. Make *cs_bar_dac* and *fs* low

      ii. Write data into *temp_dac_data*

      iii. Make *cs_bar_dac* and *fs* high

   (c) Testing your code: Make the connections as shown in Fig. 3.

   (d) Test the analog output for the digital values corresponding to the test cases mentioned in the homework part 2.

   (e) Write a program to generate a symmetrical triangular signal that varies from 0 to 3.3V with a period of 12sec (6sec+6sec)

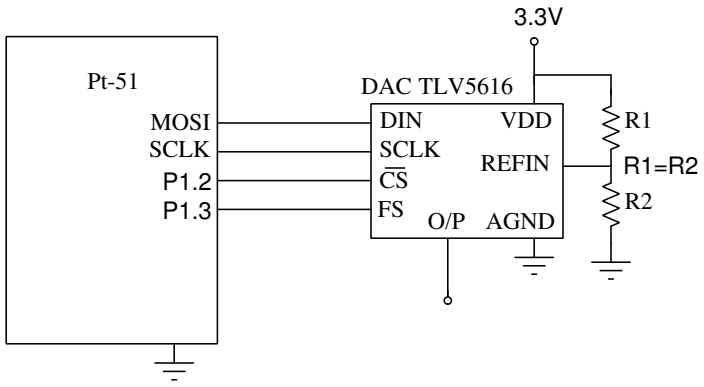

Figure 3: DAC connection diagram.

2. **Interfacing ADC and DAC together**
   Make the connections as shown in Figure 4. This is an application where a master (Pt-51) is connected to two SPI slaves (ADC and DAC). For this we will require two Slave Select instead of one.

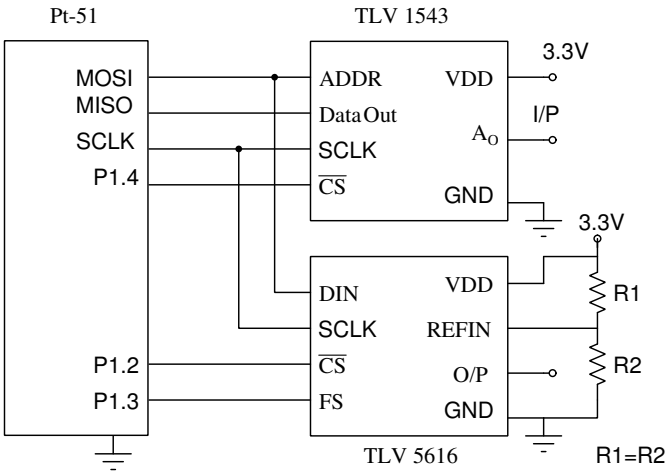

Figure 4: ADC-DAC circuit diagram

You need to make the following changes for making the entire system work:

(a) Make changes to functions **adc()** (in tlv1543.h) and **dac()** (in tlv5616.h) such that the SPCON register is changed to their respective values for each peripheral (*spi_init()* need not be changed)

(b) Make changes to **main()** function such that data is read from adc() and written to dac()

(Note that tlv1543 is 10bit ADC and tlv5616 is a 12bit DAC. How will you deal with this ?)

(c) Use sine wave (5Hz to 600Hz, 0 to 3V) as input to ADC and observe the output from DAC on the DSO.

3. **Implementing Moving Average filter: A DSP application**

   Moving average filter is simple finite length filter that can be used to remove noise. It can be implemented by computing the average of N successive samples of the input and is given by the following equation:

$$y[k] = \frac{\sum_{n=0}^{N-1} x[k-n]}{N}$$

   Example, for N=4

$$y[k] = \frac{x[k] + x[k-1] + x[k-2] + x[k-3]}{4}$$

   It can be implemented in a computationally efficient way by successively updating the moving average from the previous instance using the following equation:

$$y[k] = y[k-1] + \frac{x[k] - x[k-N]}{N}$$

   (a) Make changes to main()
      i. Take data from ADC
      ii. Use the function **moving_avg()** (in filters.h) to obtain the moving average
      iii. Give data to DAC

   (b) Plot the frequency response (magnitude only) for the above filter by sweeping the input frequency from 5Hz to 600Hz.

Function descriptions

This section describes all the functions required to be written for this interfacing experiment. 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. *main.c* have to be changed according to the requiremnet in the problem statement. Outline of the changes to be made for each part of the problem statement is as follows:

Part 1: It calls initialization functions for all peripherals which are *spi_init()* (for SPI module of Microcontroller AT89C5131), *dac_init()*(for DAC TLV5616)

1(d) Call *dac()* function for communicating with the DAC. Pass as parameters the digital values that are given in homework and verify the output voltage.

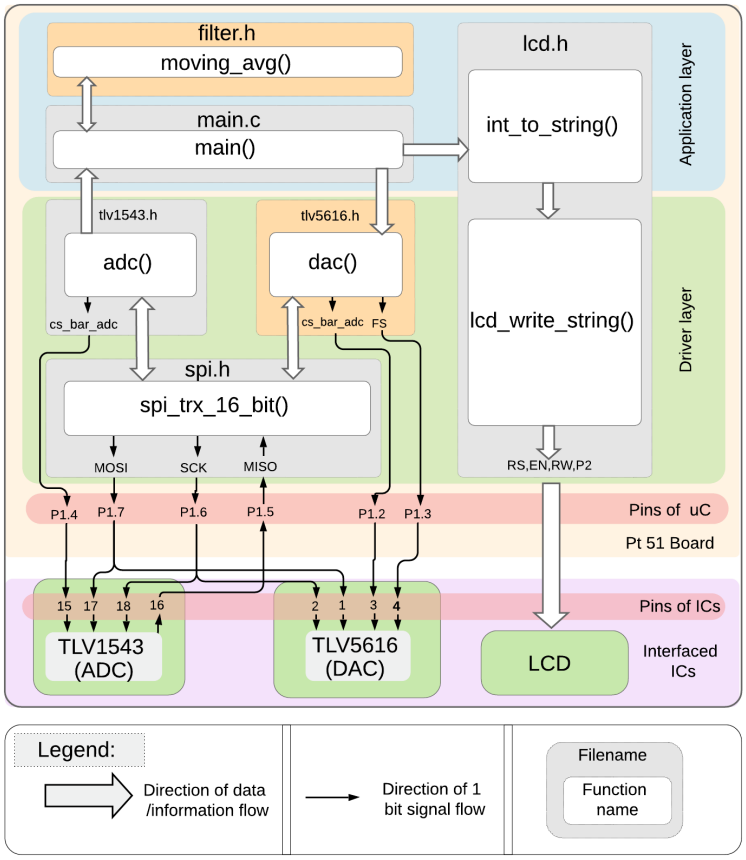

Figure 5: Software Architecture

1(e) These steps are repeated infinitely using while loop so that triangle wave in question 1(e) will be generated continuously:

Call *dac()* function for communicating with the DAC and pass as argument digital values:

i. If the ramp is in the increasing cycle, the digital value have to be incremented after appropriate dalay and passed as argument.

ii. If the ramp is in the decreasing cycle, the digital value have to be decremented after appropriate dalay and passed as argument.

Part 2: It calls initialization functions for all peripherals which are *spi_init()* (for SPI module of Microcontroller AT89C5131), *dac_init()* (for DAC TLV5616) and *adc_init()* (for ADC IC TLV1543)

These steps are repeated infinitely using while loop so that the input given to ADC will be sampled and passed to DAC continuously

i. The data returned by adc() is read

ii. The digital data is passed to dac()

Part 3: It calls initialization functions for all peripherals which are *spi_init()* (for SPI module of Microcontroller AT89C5131), *dac_init()* (for DAC TLV5616) and *adc_init()* (for ADC IC TLV1543)

These steps are repeated infinitely using while loop so that the input given to ADC will be sampled and passed to DAC continuously

i. The data returned by adc() is read

ii. The read data is passed as argument to moving_avg() and the return value from it is stored

iii. The digital data returned after moving average calculation is passed to dac()

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 platform, 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. **tlv5616.h:**
Functions in this file are specific to TLV5616. These functions control and enable data transfer between microcontroller and DAC 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 *tlv5616.h*. Configuration parameters specific to TLV5616 are set in function in this file. Following are functions in this file.

(a) *dac_init()*:

i. Parameters to be passed: None

ii. Parameters to be returned: None

iii. Task:
Control signals related to TLV5616 are initialized here.

(b) *dac()*:

i. Parameters to be passed: unsigned int - two bytes of data corresponding to the DAC analog output.

ii. Parameters to be returned: None

iii. Task:
This function triggers communication between microcontroller and DAC 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 *dac()*).

2) Send appropriate control signals for starting communication to DAC slave (DAC Chip select,frame sync etc).

3) A two byte data frame having 4 control bits and 12 DAC output value bits is formed from 16 bit data word passed to this function from other function (Check TLV5616 datasheet for more information).Select appropriate control bits. 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 DAC slave and return garbage data word since DAC do not return any value. Flush returned data word.

5) Send appropriate control signals for stopping communication to DAC slave (DAC Chip select,frame sync etc).

5. **filters.h:**
Function in this file is used to compute the moving average of samples passed as argument. It is in this function that the number of samples that is used to compute the moving average is specified (*f_no_samples*). We have specified it to be eight, but this can changed according to requirement.

moving_avg():

i. Parameters to be passed: Current Sample

ii. Parameters to be returned: Moving average of N samples

iii. Task: Computing the moving average for the specified number of samples