A to D and D to A conversion using Pts. 51

When you need to acquire a data by using the internal ADC and also output it to an external DAC using SPI protocol without buffering, we must ensure two things.

1. The sampling frequency of ADC is sufficiently high (above Nyquist) so that its sampling faithfully
2. The SPI module must be able to communicate the entire data during the inter-sampling interval.
   Here we will discuss for a input sine wave of 1KHz or below, how to set the sampling rate and also the SPI clock rate.

   The ADC in PtS51 card is having a maximum sampling rate of 500KSps.Being said this , our DAC module supports an SPI clock rate of maximum 20MHz.

Now let the system clock frequency be configured to 3 MHz. Let us choose the sampling rate needed to be 20KHz.(Good enough for low frequency signals below 1 KHz, oversampling done here).

Our DAC MCP 4901 is a 12 bit DAC. It requires 4 configuration bits along with 12 data bits to be sent via SPI for operating correctly. This implies that in a sampling interval of 50uS (for 20KHz), we should be able to send 16 bits via SPI , if we need to keep the integrity of data. This implies that the SPI clock should be atleast 16 times greater than the ADC sampling rate. So let's fix it be 750 KHz.

Steps in configuring ADC

1. As the initial step, let's choose the system clock frequency .Configure the OSCICN register to get a system clock frequency of 3 MHz.
2. Next question is how to set the sampling rate for ADC. The ADC in PtS51 can be started for conversion in many ways. One of the method is by using a software control , by setting a bit and other is by making use of Timer overflow. The first method is discussed in manual in section for ADC. But here we are going to use the second method, ie, initiating a conversion using the timer overflows.
3. The conversion time is another concern, so we need to keep the conversion clock maximum, so that a very little time is taken for data conversion by the SAR ADC. Let it be 500KHz.
4. The ADC in PtS51 is 10 bit ADC. The converted data hence will be in 2 registers, the higher byte in ADC0H and lower byte in ADC0L.The data will be left justified or right justified as per we configure it to be.

5. The ADC can operate in either single ended mode, or differential mode. We restrict our use here to single ended mode, since our DAC cannot reproduce the negative samples if at all it's taken. Since the ADC is going to be used in single ended mode of operation, make sure that the inputs given are above ground level. For sine wave add the DC offset accordingly.
6. The ADC in PtS51 has a multiplexer associated with it to choose what input needs to be connected to the ADC. The two multiplexers needs to be configured before using the ADC. The MUX AMX0P decides what needs to be connected to the positive terminal of ADC. Make sure you configure this to some port pin of your choice, where you are going to give the analog signal to ADC. The MUX AMX0N decides what needs to be connected to the negative terminal of ADC. Configure this so that the ADC is connected to GND on its negative terminal (must for single ended operation).
7. Note that if analog inputs to ADC are from a port pin ,the pin should be configured in Analog mode in PxMDIN register of the corresponding port!! Similarly if VREF is given externally to ADC that port pin also needs to be configured as analog input. All of this needs to be skipped by crossbar too. There is high probability that a mistake is committed here.
8. The VREF for the ADC has to be configured by using the REFCON register.You can configure this to be internal Vdd of the card.
9. During conversion, the AD0BUSY bit is set to logic 1 and reset to logic 0 when the conversion is complete. The falling edge of AD0BUSY triggers an interrupt (when enabled) and sets the ADC0 interrupt flag (AD0INT).When this interrupt occurs, the converted code is available in ADC0 data registers ADC0H:ADC0L.Data can be in right justified or left justified formats based on the setting of ADC0CF register.

   Registers to be configured
   1. PxMDIN to be configured to make the input port pins as analog inputs. XBR0 and XBR1 needs to be configured for enabling crossbar and also SPI module if used.
   2. ADC0CF needs to be configured for data justification and setting the conversion clock rate of ADC. Set the conversion clock rate to maximum, ie 500KHz.
   3. The ADC0CN needs to be configured for enabling the ADC and also to set the source of start of conversion signal for ADC. Configure this to be Timer0 overflow. This register is bit addressable.
   4. The REF0CN must be configured to choose the reference voltage for ADC. This you can configure to take Vdd of the card as Vref. Make sure you select VDD as VREF. REF0CN = 0x08.
   5. The AMX0P and AMX0N needs to be configured as explained earlier in the document
   6. Enable all interrupts which are not masked by setting EA.
   7. Now configure EIE1 register to enable the ADC interrupt.

Here you have chosen timer0 as the clock source. You need to configure this timer accordingly. The CKC0N register needs to be configured to choose the timer clock frequency. The timer needs to be operated in 8 bit auto reload mode. Find the reload value and load in TH0 and TL0.Start the timer. Each timer overflow will cause a start of conversion in ADC. Clear TF0 in the ADC interrupt routine, so that the timer keeps on running and generating interrupts at every 50uS corresponding to 20KHz sampling rate.

Note : When using SPI module and ADC together make sure to keep track of ports used as analog inputs, the port pins used for SPI and avoid conflicts by correctly configuring PnSKIP registers. You can refer to manual on how to do this on port configuration section.

SPI communication to MCP4921 DAC

As we discussed in the starting of the document, we are planning to have a SPI clock frequency of 750KHz. Here since we are using the MCP4921 DAC, it doesn't send data back to the controller. This implies that the MISO pin of PtS51 can be just tied to gnd or left open. The NSS pin represents the chip select and the MOSI is used for transmitting data to the DAC. The LDAC pin of the DAC can be either grounded if we don't need synchronous output, or should be tied to a port pin which will give a pulse at the end of the data transfer. Please refer the MCP4921 datasheet and study the serial communication section of DAC thoroughly. This section proceeds with an idea that you have read and understood the MCP4921 datasheet.

In PtS51, we have an SPI module. SPI in PtS51 can be operated in 4 wire/ 3 wire modes. It can operate in multi master/single master environment. Here in our problem, only PtS51 is the master and the MCP4921 device is slave, ie, the mode of operation is single master, 4 wire SPI.

As you know from reading the datasheet of MCP4921, it needs data to be clocked in on the rising edge of the clock, or in other words the controller needs to output the data on the SPI bus at the falling edge of the clock. In SPI, clock polarity and phase determines at what clock edge data is put in the bus and what clock edge it gets sampled. We should ensure that the clock polarity and phase match between slave and master for faithful communication.
Suppose we needs our clock to be idle in high state. We say the CLOCK PHASE = 1. We know from the datasheet of MCP4921 that it is going to sample the data on the rising edge of clock. This implies that the clock needs to go low initially and on the next rising edge data gets sampled, in other words CLOCK PHASE =1.We needs to configure this for SPI to work correctly.

The various registers and their configuration for SPI communication are as follows

1. Enable the SPI bus and crossbar by configuring the XBR0 and XBR1 registers.
2. Enable the single master mode and configure the clock phase and polarity in SPI0CFG register
3. Set the SPI clock frequency in SPI0CKR register. Make sure to set it 750KHz for a system clock of 3MHz.

4. Set NSSMD1 bit in SPI0CN , to operate in single master 4 wire mode
5. Set the initial value of chip select in NSSMD0 in SPI0CN. Here it is high.

A SPI master device initiates all data transfers on a SPI bus. SPI0 is placed in master mode by setting the Master Enable flag (MSTEN, SPI0CN.6). Writing a byte of data to the SPI0 data register (SPI0DAT) when in master mode writes to the transmit buffer. If the SPI shift register is empty, the byte in the transmit buffer is moved to the shift register, and a data transfer begins. The SPI0 master immediately shifts out the data serially on the MOSI line while providing the serial clock on SCK. The SPIF (SPI0CN.7) flag is set to logic 1 at the end of the transfer. If interrupts are enabled, an interrupt request is generated when the SPIF flag is set.
In our case, we can write the data received from ADC registers to SPI0DAT. Make sure you frame the data 10 bit data received, make it as 12 bits, add 4 configuration bits for DAC and sent it. First you need to clock out the control bits followed by MSB of data and so on. When we write data to SPI0DAT, it gets shifted out with MSB first.

6. After a byte of data gets written into SPI0DAT, make sure that another one doesn't get overwritten unless the transfer is complete, ie SPI0DAT is empty. This can be done in multiple ways, by checking the status of SPIF and so on. Do it as per your logic. You can use interrupts if needed.
7. If the LDAC pin is tied to ground, the data sent by the controller to DAC gets latched in as soon as 16 bits have been sent and the DAC gives the corresponding output.
8. The chip select (NSS pin of controller) should be brought low before data transmission and should be made high as soon as 16 bits are transmitted.

   Please make it a point to go through the datasheet of MCP4921.

   Example:
   Given we need to use internal ADC to convert a signal of frequency 1KHZ into digital values and sent the digital code obtained to DAC using SPI.

   Let's choose system clock = 3MHz. Configure the OSCICN accordingly.
   Now let the sampling rate be 20KHz. So the ADC has to start conversion by a timer overflow at intervals of 50us. Configure timer 0 as 8 bit autoreload mode and find the value to be loaded in the timer registers.{ select the timer clock prescale factor correctly in CKC0N}

Let the data be left justified. Now we need the conversion time to be minimum. Configure ADC0CF register for that purpose.

REFC0N to configure the VREF of ADC
AMX0P and AMX0N to configure the ADC input pins. Make sure the ADC input pins are configured as analog inputs.
ADC0CN to enable ADC , EA to enable interrupts globally and EIE to enable ADC interrupts. Configure this registers accordingly.
Now select the SPI clock frequency in SPI0CKR
Enable the crossbar and SPI module by configuring XBR registers.
Configure SPI0CFG and SPI0CN registers as explained before.
Write data to SPI0DAT for transmission .Write only when SPI0DAT is empty. You can check weather a transmission is complete using the status of SPIF bit.

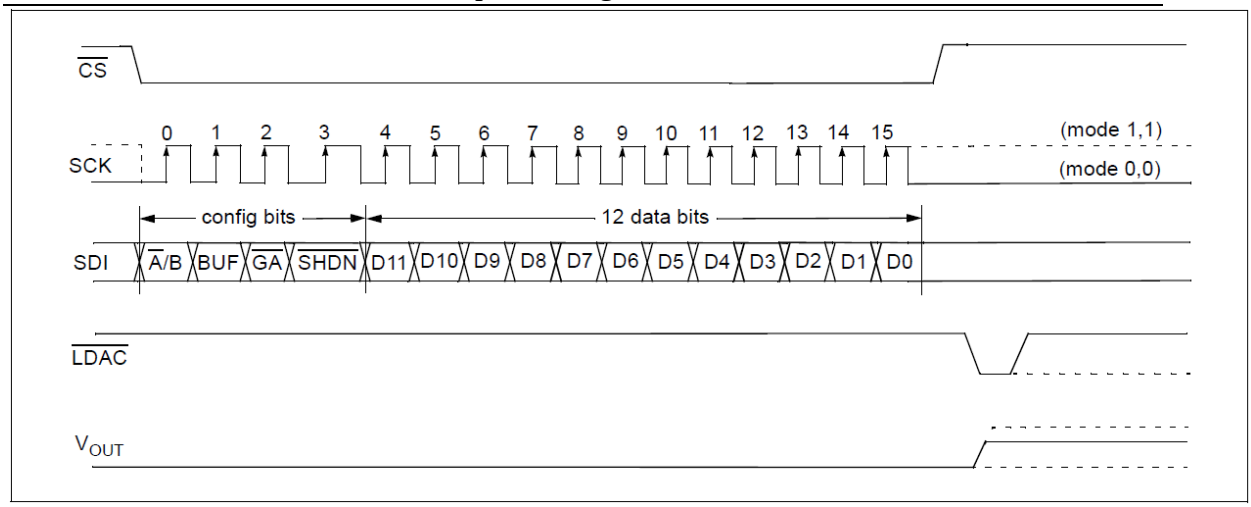

Fig : Write command to DAC. See how the data is transmitted and the CS signal.

REGISTER : WRITE COMMAND REGISTER

Upper Half:							
W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x
$\overline{A/B}$	BUF	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
bit 15							bit 8

Lower Half:							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
bit 7							bit 0

bit 15 **$\overline{A/B}$:** DAC_A or DAC_B Select bit
1 = Write to DAC_B
0 = Write to DAC_A

bit 14 **BUF:** V_{REF} Input Buffer Control bit
1 = Buffered
0 = Unbuffered

bit 13 **$\overline{GA}$:** Output Gain Select bit
1 = 1x ($V_{OUT} = V_{REF} * D/4096$)
0 = 2x ($V_{OUT} = 2 * V_{REF} * D/4096$)

bit 12 **$\overline{SHDN}$:** Output Power Down Control bit
1 = Output Power Down Control bit
0 = Output buffer disabled, Output is high impedance

bit 11-0 **D11:D0:** DAC Data bits
12 bit number "D" which sets the output value. Contains a value between 0 and 4095.

Fig : Write command register of DAC