Lab 4B - DAC Interfacing

Wadhwani Electronics Lab
Compiled by: Devdatta and Jishnu

Department of Electrical Engineering
Indian Institute of Technology Bombay
January 9, 2016

Problem Statements

- Understanding Serial Peripheral Interface (SPI).
- Getting familiar with Microchip's MCP4921 Serial DAC IC.
- Give an analog signal as input to the ADC pin and produce the same signal at the output of DAC.

Instructions: 1 (Setting up SSI)

1. Please read the document provided for SPI Communication first and then proceed ahead.
2. Find the DAC datasheet here : MCP4921 Datasheet.
3. The DAC uses SPI interface. Include "driverlib/ssi.h" in your *.c* file.
4. Enable one the SSI modules using the function SysCtlPeriphEnable(), Page 505.

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
SSI0Clk	19	PA2 (2)	I/O	TTL	SSI module 0 clock
SSI0Fss	20	PA3 (2)	I/O	TTL	SSI module 0 frame signal
SSI0Rx	21	PA4 (2)	I	TTL	SSI module 0 receive
SSI0Tx	22	PA5 (2)	O	TTL	SSI module 0 transmit

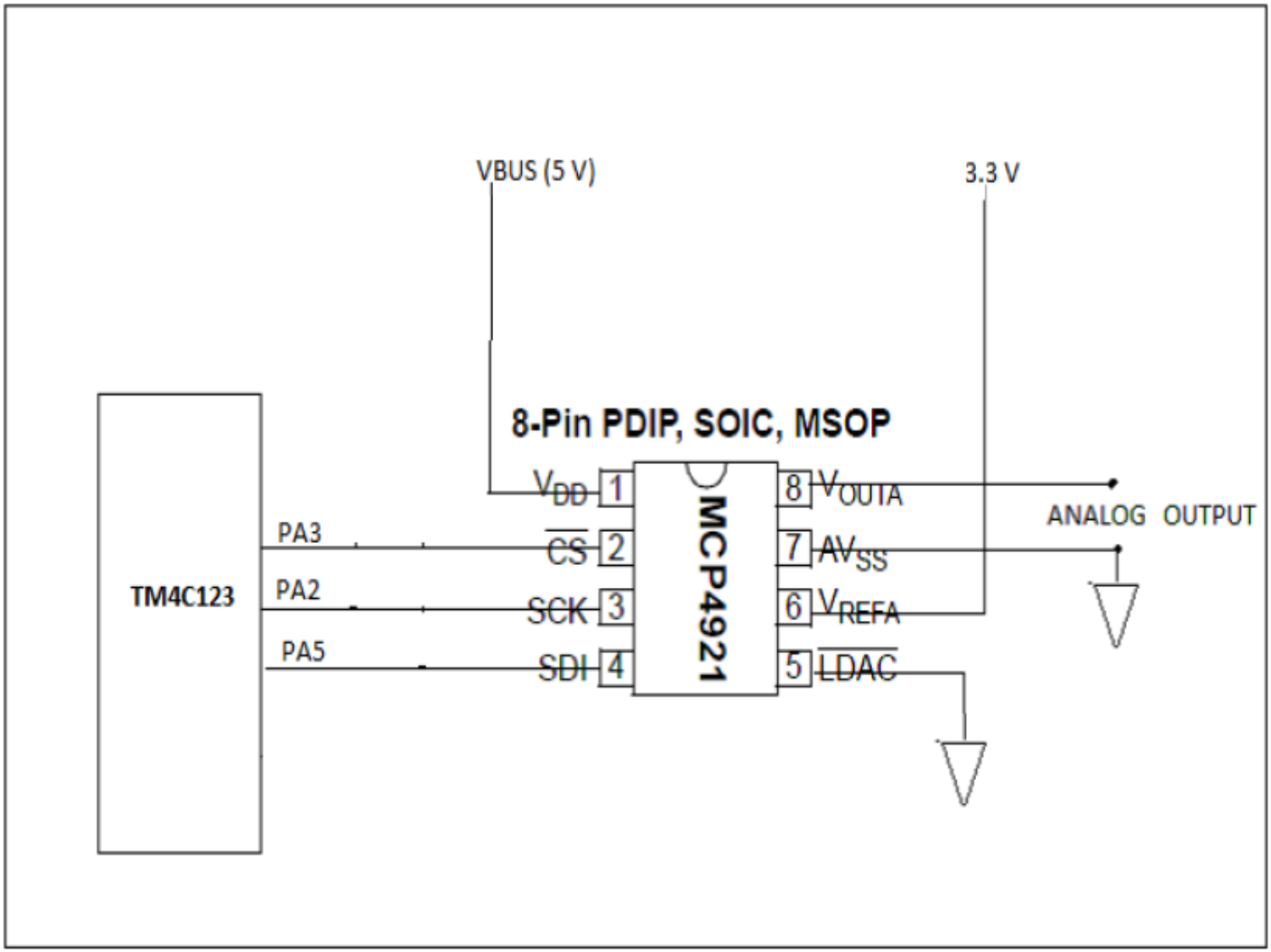
5. Set the pin type of the of the associated pins as **SSI**. Use GPIOPinTypeSSI(), Page 281 for this.
6. Configure the SSI with appropriate parameters. Use SSIConfigSetExpClk(), Page 469 for this. (Hint : Use SysCtlClockGet(), Page 488 for system clock.)
7. Enable the SSI using the function SSIEnable(), Page 473.

Instructions: 2 (Setting up ADC)

- Set up the ADC as you did in Exp 4A.
- Here we need to set up a pin as ADC input. Check the TM4C123GH6PM datasheet, Page 801 to find which pin can be configured as ADC input. Use GPIOPinTypeADC(), Page 266 to set.
- Modify the ADCSequenceStepConfigure(), Page 42 function so as to take input from the ADC channel of input pin (instead of the Temperature Sensor)

Instructions: 3 (Connections)

Interfacing MCP4921 to TM4C123



Instructions: 4 (Code Algorithm)

- Trigger the ADC using ADCProcessorTrigger(), Page 36. Wait while the ADC is busy(Use ADCBusy(), Page 23).
- Read the value from ADC using ADCSequenceDataGet(), Page 39.
- Construct the appropriate frame for DAC as given below:

bit 15	**$\overline{\text{A/B}}$:** 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	**$\overline{\text{GA}}$:** Output Gain Select bit 1 = 1x ($V_{OUT} = V_{REF} * D/4096$) 0 = 2x ($V_{OUT} = 2 * V_{REF} * D/4096$)
bit 12	**$\overline{\text{SHDN}}$:** 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.

- Send the frame to DAC by SSIDataPut(), Page 471. Wait till SSI is busy. Check using SSIBusy(), Page 467.

Tips and Precautions

- Make sure you have included the following header files: "stdint.h", "stdbool.h", "inc/hw_memmap.h", "inc/hw_ssi.h", "inc/hw_types.h", "driverlib/gpio.h", "driverlib/sysctl.h", "driverlib/pin_map.h", "driverlib/adc.h" and "driverlib/ssi.h".
- Make sure the **entire** waveform you give to the ADC is **POSITIVE**. Provide appropriate DC bias to ensure that the input is in range.
- This DAC accepts 16-bit digital data as input. Make sure you pad the ADC data with proper control bits before sending it to DAC.