

Name _____

Lab Section _____

PIC – Serial Peripheral Interface (SPI) to Digital Pot

Lab 7

Introduction: SPI is a popular synchronous serial communication protocol that allows ICs to communicate over short distances (PCB level communication). In this lab you will use the Serial Peripheral Interface (SPI) to communicate with a digital potentiometer.

Lab Requirements:

1. Demonstration of a digital potentiometer as a programmable attenuator.
2. Submission of your neatly formatted source code.

Demo Check (JK)_____

About the Serial Peripheral Interface: SPI is a full-duplex synchronous serial communication protocol for data transfer between integrated circuits on a PCB. The protocol uses a master/slave relationship where the master initiates all data transfers and generates the data synchronization clock. The standard bus connections are a Serial Clock (SCK), Master-Out-Slave-In (MOSI), Master-In-Slave-Out (MISO) and an active low Slave Select (nSS). Each slave device has its own Slave Select (nSS) signal which dictates when the device is participating in bus transfers. A typical SPI connection diagram is shown below:

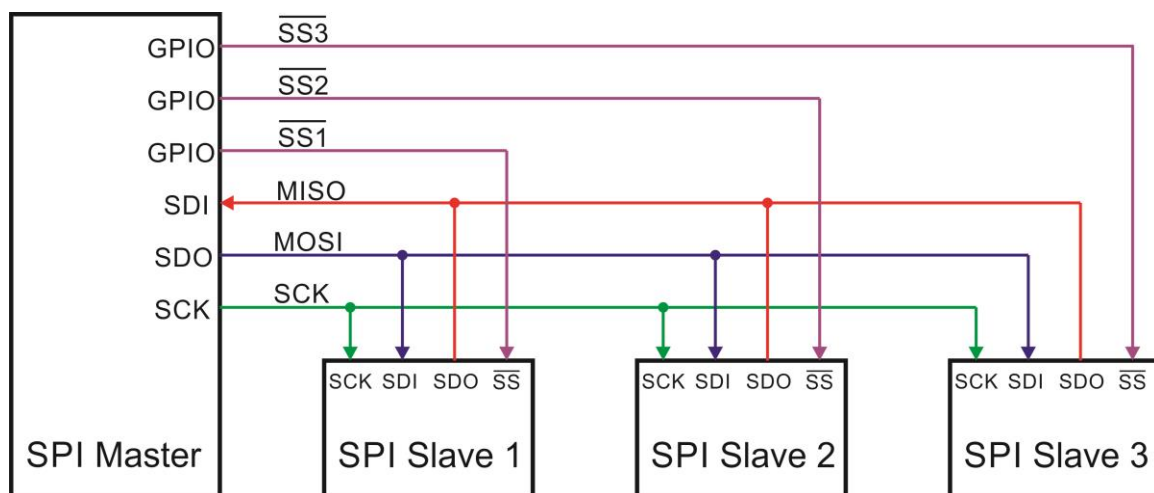


Figure 1 - SPI Connection Diagram

The SPI protocol operates as a shift register where data is transferred from the master to the slave and from the slave to the master at the same time. The data is shifted one bit at a time on the SCK edge. Unfortunately, there is no standardization on the clock edge for data shift or on the clock polarity so four different modes are possible. When connecting to a new peripheral it is important to study the waveform timing diagram to determine the clock polarity and phase.

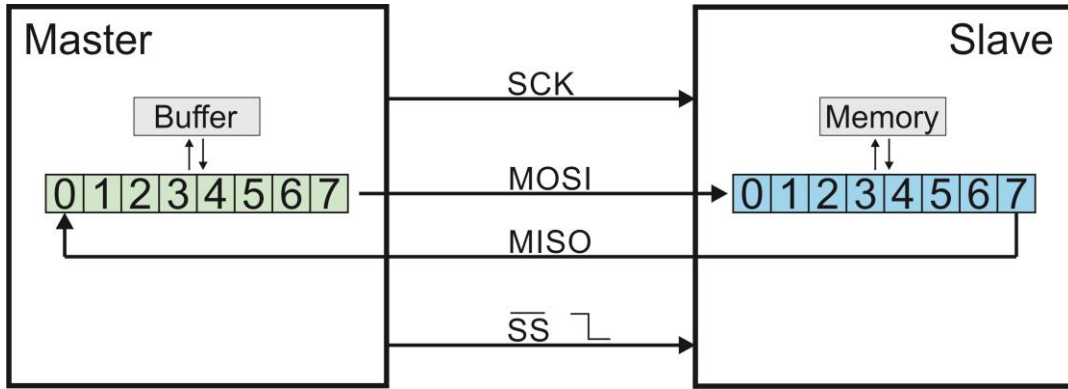


Figure 2 - SPI Shift Register

Before a data transfer begins the master must assert the nSS signal by driving it low. Then the data will be exchanged one bit at a time between the master and the slave. The Data Transfer diagram below illustrates a Mode 3 transfer where the SCK idles high and data is valid on the rising edge of the SCK. At the end of the data transfer the nSS signal is brought high to deselect the peripheral.

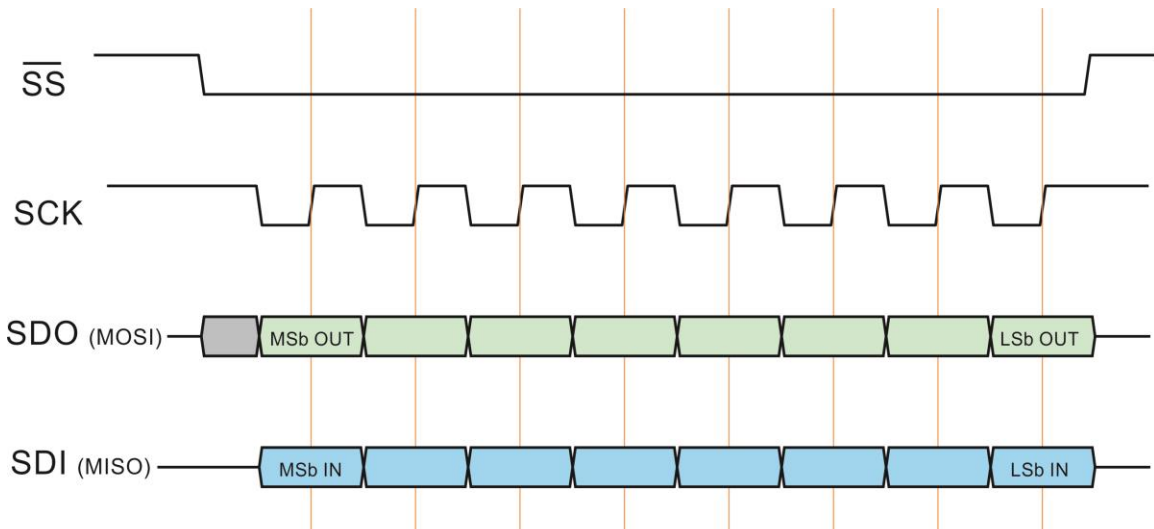


Figure 3 - SPI Data Transfer

MODE 3

Most modern microcontrollers contain dedicated hardware to implement a master or slave peripheral interface. In the PIC family of microcontrollers this functionality is located in the Master Synchronous Serial Port (MSSP) Module. The MSSP is capable of being configured to implement either the SPI or I2C bus protocol. The hardware can be configured to operate as either a master or as a slave device. Today we will be setting up the MSSP for SPI Master Mode operation.

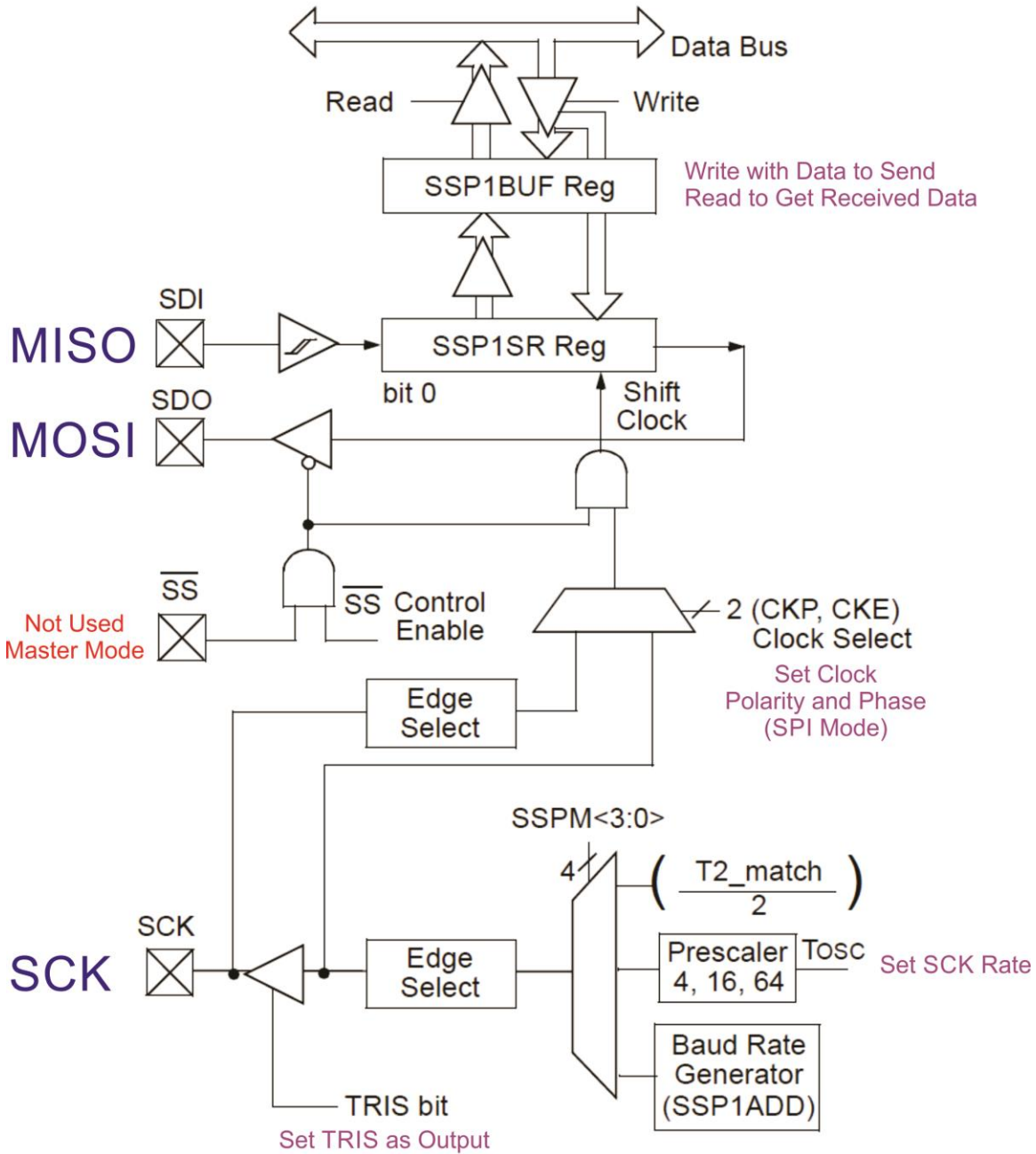


Figure 4 - MSSP SPI Block Diagram

REGISTER 29-2: SSP1CON1: SSP CONTROL REGISTER 1

R/C/HS-0/0	R/C/HS-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
WCOL	SSPOV ⁽¹⁾	SSPEN	CKP	SSPM<3:0>			
bit 7							bit 0

Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HS = Bit is set by hardware C = User cleared

bit 7	WCOL: Write Collision Detect bit (Transmit mode only) 1 = The SSP1BUF register is written while it is still transmitting the previous word (must be cleared in software) 0 = No collision
bit 6	SSPOV: Receive Overflow Indicator bit ⁽¹⁾ <u>In SPI mode:</u> 1 = A new byte is received while the SSP1BUF register is still holding the previous data. In case of overflow, the data in SSP1SR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSP1BUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSP1BUF register (must be cleared in software). 0 = No overflow <u>In I²C mode:</u> 1 = A byte is received while the SSP1BUF register is still holding the previous byte. SSPOV is a "don't care" in Transmit mode (must be cleared in software). 0 = No overflow
bit 5	SSPEN: Synchronous Serial Port Enable bit In both modes, when enabled, the following pins must be properly configured as input or output <u>In SPI mode:</u> 1 = Enables serial port and configures SCK, SDO, SDI and \overline{SS} as the source of the serial port pins ⁽²⁾ 0 = Disables serial port and configures these pins as I/O port pins <u>In I²C™ mode:</u> 1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins ⁽³⁾ 0 = Disables serial port and configures these pins as I/O port pins
bit 4	CKP: Clock Polarity Select bit <u>In SPI mode:</u> 1 = Idle state for clock is a high level 0 = Idle state for clock is a low level <u>In I²C™ Slave mode:</u> SCL release control 1 = Enable clock 0 = Holds clock low (clock stretch). (Used to ensure data setup time.) <u>In I²C™ Master mode:</u> Unused in this mode
bit 3-0	SSPM<3:0>: Synchronous Serial Port Mode Select bits 1111 = I ² C™ Slave mode, 10-bit address with Start and Stop bit interrupts enabled 1110 = I ² C™ Slave mode, 7-bit address with Start and Stop bit interrupts enabled 1101 = Reserved 1100 = Reserved 1011 = I ² C™ firmware controlled Master mode (slave idle) 1010 = SPI Master mode, clock = FOSC/(4 * (SSP1ADD+1)) ⁽⁵⁾ 1001 = Reserved 1000 = I ² C™ Master mode, clock = FOSC / (4 * (SSP1ADD+1)) ⁽⁴⁾ 0111 = I ² C™ Slave mode, 10-bit address 0110 = I ² C™ Slave mode, 7-bit address 0101 = SPI Slave mode, clock = SCK pin, \overline{SS} pin control disabled, \overline{SS} can be used as I/O pin 0100 = SPI Slave mode, clock = SCK pin, \overline{SS} pin control enabled 0011 = SPI Master mode, clock = T2_match/2 0010 = SPI Master mode, clock = FOSC/64 0001 = SPI Master mode, clock = FOSC/16 0000 = SPI Master mode, clock = FOSC/4

SSP1CON1 = 0b00110010;

To exchange data between the master and slave simply load the SSP data buffer then block until the transfer is finished. The code below also clears collision flag just in case it was set by a poorly timed write to the buffer. Don't forget assert the nSS line before calling the `SPI_SHIFT_8` function.

```
uint8_t SPI_SHIFT_8 (uint8_t data)
{
    SSP1CON1bits.WCOL = 0;           // Clear Write Collision flag just in Case
    SSP1BUF = data;                 // Load Buffer with Data to Shift
    while (SSP1STATbits.BF == 0){} // Block until 8b transferred
    return (SSP1BUF);               // Return Data/Dummy
}
```

Digital Potentiometer: Digital potentiometers can be useful in circuits where you need to control an analog function with a microcontroller. For this lab you will be interfacing the PIC16F18324 to a MCP4151-503 (50k) Potentiometer using the SPI Interface. The MCP4151 is available in an 8-pin DIP package which forces an unusual SPI interface due to the low pin count.

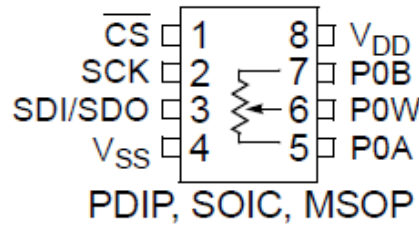


Figure 5 - MCP4151 Pinout

The MCP4151 uses a shared bi-directional SDI/SDO line in order to fit both pins on the small package. R1 must be sized to not limit the SDO voltage below the logic threshold of the SDI of the MCP4151. Since there is not much value in reading the registers of the digital pot we will simplify our design by not using the MISO (SDI) connection.

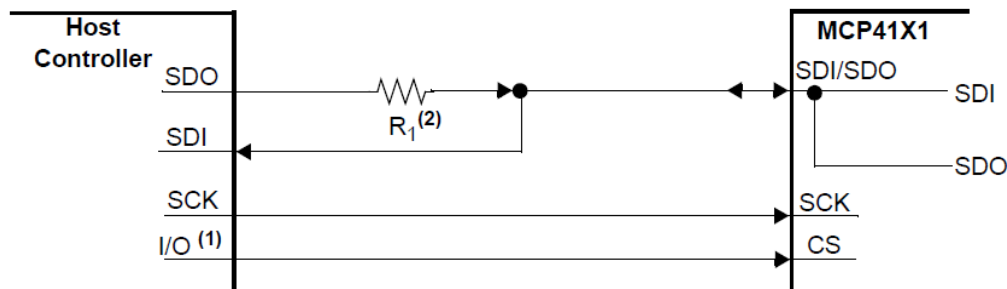


Figure 6 – MCP4151 Bi-Directional Hardware Configuration

If we only care about setting the digital potentiometers wiper position we can simplify the connection as shown in figure 7 below. If we use this simplified method to interconnect the parts we must be careful to only issue write commands from the master.

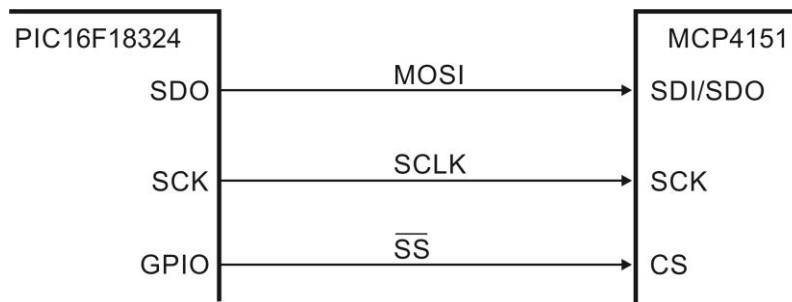


Figure 7 - Simplified Connection Diagram

Data packets can be either 8 or 16 bits in length depending on the function. The general memory map for Microchip digital potentiometers is shown in Figure 9 below. For the MCP4151 we will only be writing to address **00h** which will set the wiper position of the potentiometer. The pot has 257 steps so the range of values to be written to the data transfer payload is 0 to 256. A convenient way to create an input that represents the desired wiper position is to use the analog to digital converter connected to the trim pot on your board. The 10b ADC value will need to be scaled and manipulated in order to create the 8b +1 range of the digital pot.

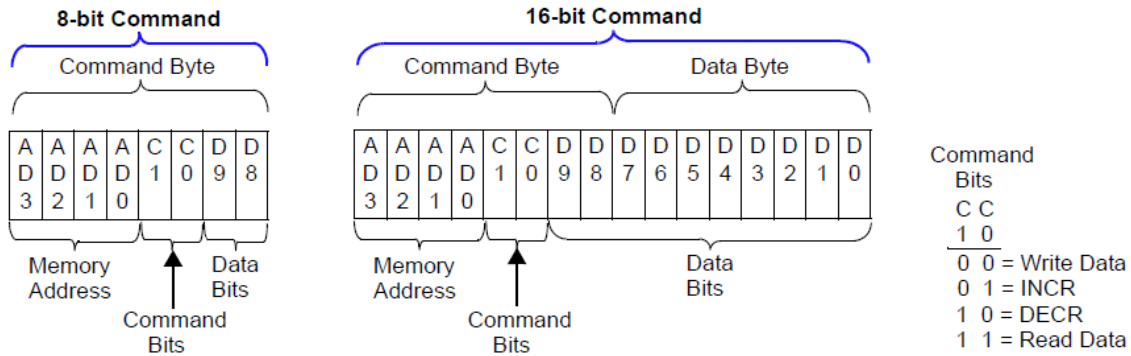


Figure 8 - MCP4151 Commands

TABLE 7-2: MEMORY MAP AND THE SUPPORTED COMMANDS

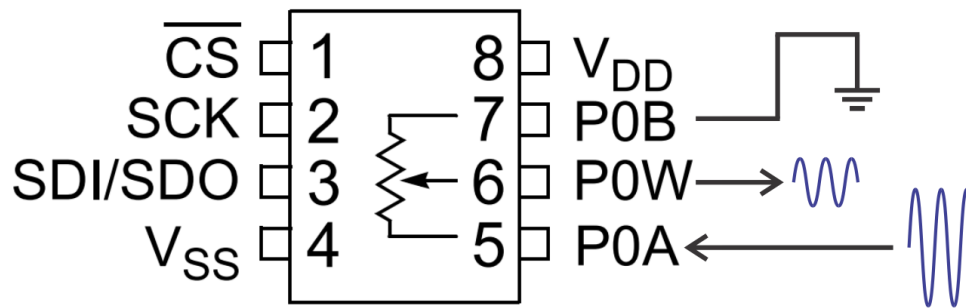
Address		Command	Data (10-bits) ⁽¹⁾	SPI String (Binary)	
Value	Function			MOSI (SDI pin)	MISO (SDO pin) ⁽²⁾
00h	Volatile Wiper 0	Write Data	nn nnnn nnnn	0000 00nn nnnn nnnn	1111 1111 1111 1111
		Read Data	nn nnnn nnnn	0000 11nn nnnn nnnn	1111 111n nnnn nnnn
		Increment Wiper	—	0000 0100	1111 1111
		Decrement Wiper	—	0000 1000	1111 1111
01h	Volatile Wiper 1	Write Data	nn nnnn nnnn	0001 00nn nnnn nnnn	1111 1111 1111 1111
		Read Data	nn nnnn nnnn	0001 11nn nnnn nnnn	1111 111n nnnn nnnn
		Increment Wiper	—	0001 0100	1111 1111
		Decrement Wiper	—	0001 1000	1111 1111
02h	Reserved	—	—	—	—
03h	Reserved	—	—	—	—
04h	Volatile TCON Register	Write Data	nn nnnn nnnn	0100 00nn nnnn nnnn	1111 1111 1111 1111
		Read Data	nn nnnn nnnn	0100 11nn nnnn nnnn	1111 111n nnnn nnnn
05h	Status Register	Read Data	nn nnnn nnnn	0101 11nn nnnn nnnn	1111 111n nnnn nnnn
06h-0Fh	Reserved	—	—	—	—

Note 1: The Data Memory is only 9-bits wide, so the MSb is ignored by the device.
 Note 2: All these Address/Command combinations are valid, so the CMDERR bit is set. Any other Address/Command combination is a command error state and the CMDERR bit will be clear.

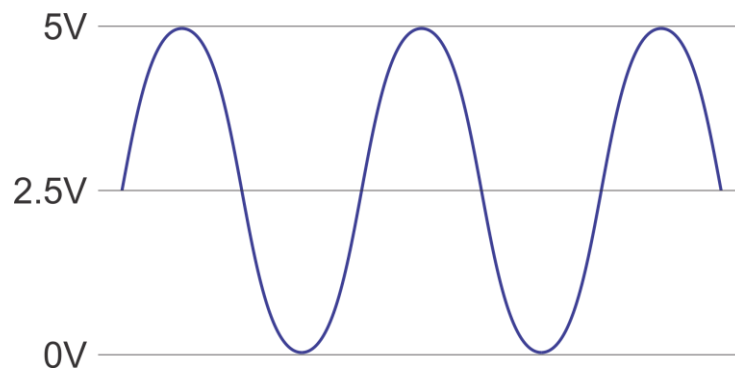
Figure 9 - MCP4151 Memory Map

Digital Potentiometer Interfacing: Unlike a traditional passive potentiometer, the digital potentiometer can only work with signals that fall within the power rails of the chip. Signals outside of the power rails (0 to 5V for this lab) can permanently damage the component and must be avoided.

To build the programmable attenuator you will configure the digital pot as a voltage divider by feeding the signal to be attenuated into the P0A pin and grounding the P0B pin. The output will be taken from the wiper at P0W.



You will use a signal generator to create a 1kHz Sine wave with a 5Vpp amplitude and 2.5V of DC offset. Verify the signal amplitude and offset using the oscilloscope before connecting it to the digital pot.



Schematic: Neatly draw the schematic of your circuit in the box below. Be sure to include component values, pin numbers, supply voltages and support circuitry. You do not need to include unused header connections or the microcontroller's ICSP interface.

