Digital-to-Analog Conversion

Table of Contents

- **Microchip MCP4921 Features**
- **General Overview**
- **Serial SPI Interface**
  - Overview
  - Programmable Gain Block (GA)
  - Voltage Reference Amplifiers (BUF)
- **Arduino SPI Test Script**
- **Bipolar Operation**

References:

1. 8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash - ATmega328P, Chapter 23 "Analog-to-Digital Converter", ATMEL document doc8161
2. Microchip Technology Inc. MCP4921/MCP4922, 12-Bit DAC with SPI™ Interface, Microchip Technology document DS21897B

Introduction

Before we begin watch the following instructional video on Digital Audio - you may be surprised to discover that much of what you thought you knew about digitization is wrong.

http://www.wired.com/gadgetlab/2013/02/sound-smart-watch-this-excellent-primer-on-digital-audio/

Microchip MCP4921 Features

- 12-Bit Resolution
- ±0.2 LSB DNL (typ)
- ±2 LSB INL (typ)
- Rail-to-Rail Output
- SPI™ Interface with 20 MHz Clock Support
- Fast Settling Time of 4.5 µs
- Selectable Unity or 2x Gain Output
- External VREF Input
- 2.7V to 5.5V Single-Supply Operation

General Overview

The MCP4921 is a voltage output string DAC. As shown in Figure 1, this DAC includes an input amplifier, rail-to-rail output amplifier, reference buffer, plus shutdown and reset management circuitry.

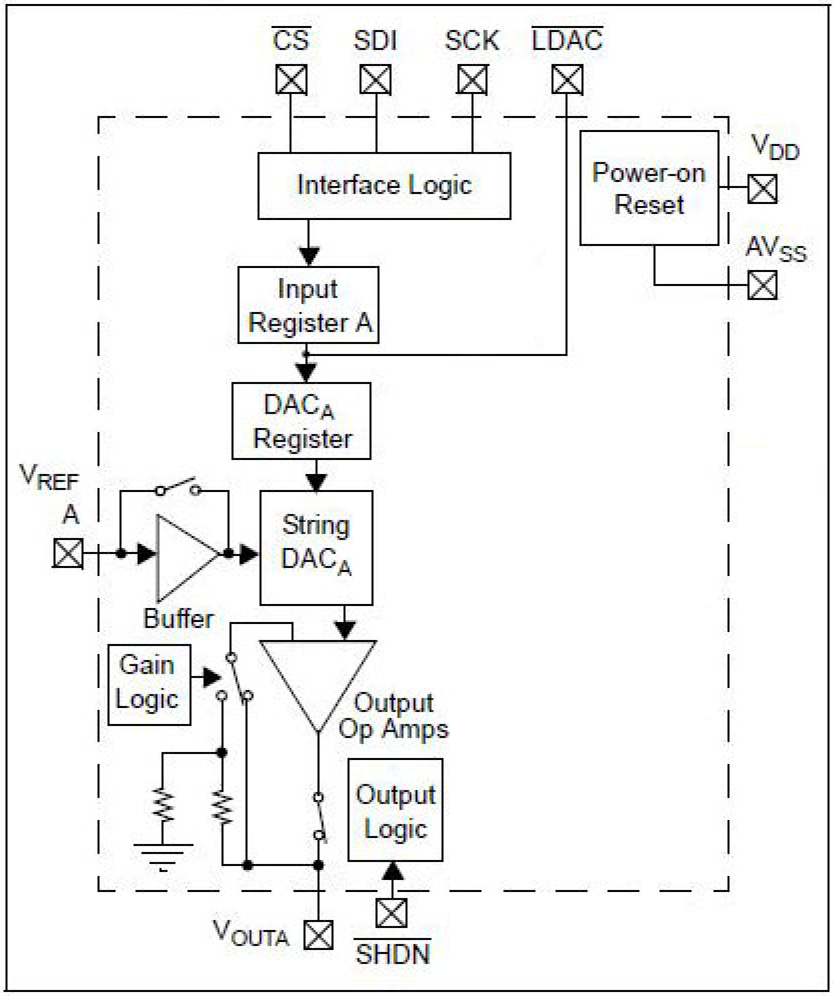


Figure 1 MCP4921 DAC Simplified Block Diagram

Serial communication conforms to the SPI protocol Modes 0 and 3 as defined by the ATmega328P datasheet. The coding of these devices is straight binary and the ideal output voltage is given by Equation 1, where G is the selected gain (1x or 2x), D_N represents the digital input value and n represents the number of bits of resolution (n = 12).

$$V_{OUT} = \frac{V_{REF} G D_N}{2^n}$$

Equation 1 LSB Size

1 LSB is the ideal voltage difference between two successive codes. Table 1 illustrates how to calculate LSB.

Device	V_{REF}, GAIN	LSB SIZE
MCP492X	External V_{REF}, 1x	$V_{REF}/4096$
MCP492X	External V_{REF}, 2x	$2\ V_{REF}/4096$

Serial SPI Interface

Overview

The MCP4921 is designed to interface directly with the Serial Peripheral Interface (SPI) port, of the ATmega328P microcontrollers as shown in Figure 2. Your circuit should include a 0.1 μF by-pass capacitor in order to filter high-frequency noise. The noise can be induced onto the power supply's traces or as a result of changes on the DAC's output. The bypass capacitor helps to minimize the effect of these noise sources on signal integrity. This capacitor should be placed as close to the device power pin (VDD) as possible (within 4 mm). The power source supplying these devices should be as clean as possible.

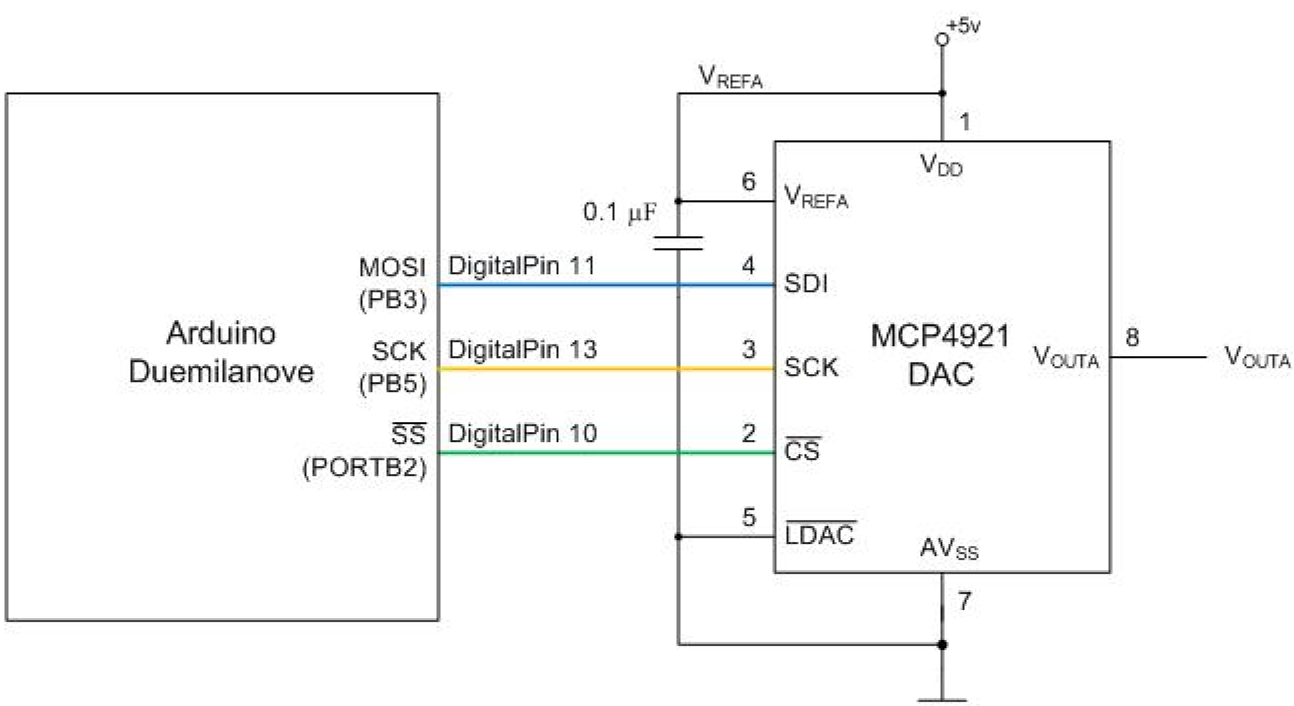


Figure 2 Arduino to MCP4921 DAC Electrical Interface

The MCP4921 is designed to interface with the Serial Peripheral Interface (SPI) port of the

ATmega328P microcontrollers in SPI Mode 0 or Mode 3 as defined in Table 2 SPI Modes (Table 18.2 of the ATmega328P datasheet). The MCP4921 datasheet refers to these as modes 0, 0 and 1, 1 respectively (see Figure 3).

SPI Mode	Conditions	Leading Edge	Trailing eDge
0	CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)
1	CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)
2	CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)
3	CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)

Table 2 ATmega328P SPI Modes

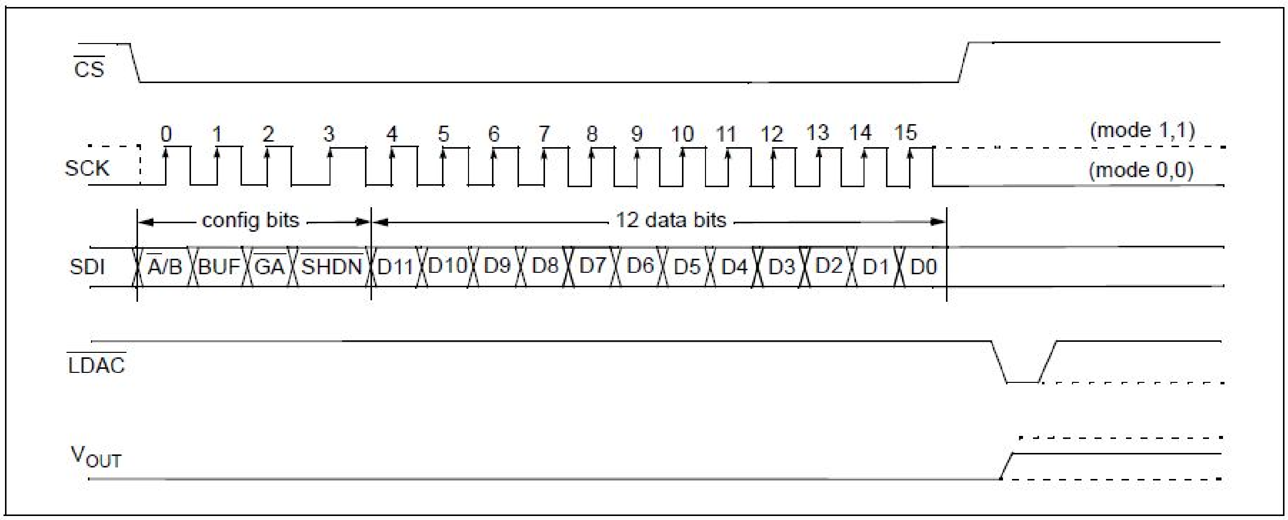


Figure 3 Writing 16-bit Command/Data Word to the MCP4921 DAC

Commands and data are sent to the device via the SDI pin, with data being clocked-in on the rising edge of SCK. The communications are unidirectional and, thus, data cannot be read out of the MCP4921. The CS pin must be held low for the duration of a write command. The write command/data consists of 16 bits and is used to configure the DAC's control and data latches. Figure 4 details the input registers used to configure and load the DACA registers.

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.

Figure 4 Writing Command/Data Word Register

For the MCP4921 you should keep A/B bit 15 at logic zero and SHDN bit 12 at logic 1. Bits 14 (BUF) and 13 (GA), may be customized to your design solution as discussed in the next two Sections.

Programmable Gain Block (GA)

The rail-to-rail output amplifier has configurable gain allowing optimal full-scale outputs for differing voltage reference inputs. The output amplifier gain has two

selections, a gain of 1 V/V (GA = 1) or a gain of 2 V/V (GA = 0). The output range is ideally 0.000V to 4095/4096 * V_{REF} when G = 1, and 0.000 to 4095/4096 * V_{REF} when G = 2. The default value for this bit is a gain of 2, yielding an ideal full-scale output of 0.000V to 4.096V when utilizing a 2.048V VREF. Note that the near rail-to-rail CMOS output buffer's ability to approach AVSS and VDD establish practical range limitations. The output swing specification in the Datasheet Section 1.0 "Electrical Characteristics" defines the range for a given load condition.

Voltage Reference Amplifiers (BUF)

The input buffer amplifiers for the MCP492X devices provide low offset voltage and low noise.

A configuration bit for each DAC allows the VREF input to bypass

the input buffer amplifiers, achieving a Buffered or Unbuffered mode. The default value for this bit is unbuffered. Buffered mode provides a very high input impedance, with only minor limitations on the input range and frequency response. Unbuffered mode provides a wide input range (0V to VDD), with a typical input impedance of 165 kΩ w/7 pF.

Arduino SPI Test Script

The following program is designed to read Arduino Analog Pin 0, and then convert it to a 10-bit integer using the ATmega328 ADC subsystem. This 10-bit value is then sent to the 12-bit MCP4921 DAC. This 10-bit value is left justified with respect to the 12-bit DAC data word. This script does not use the SPI peripheral of the ATmega328. Instead it manually shifts out the bits.

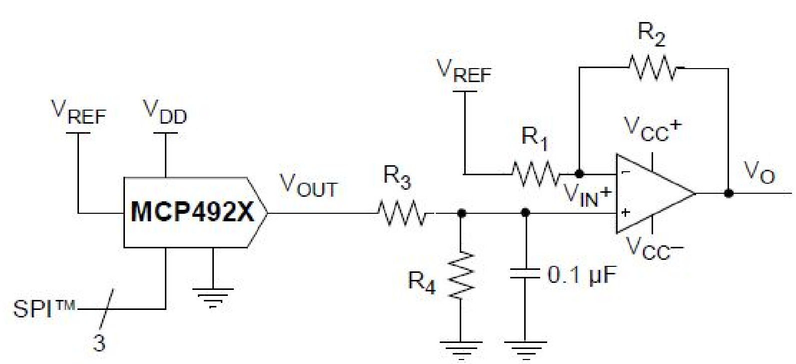
```
//***********************************************************//
//  Name         : Analog In to 12-bit SPI DAC Microchip MCP4921 //
//  Author       : Gary Hill                                 //
//  Date         : 14 March, 2010                            //
//  Version      : 1.0                                       //
//  Reference(s): http://www.arduino.cc/playground/Code/Spi  //
//  Notes        : Download Spi folder and place in          //
//               : arduino-00nn/hardware/libraries folder    //
//*************************************************************

// SPI Interface     SS_PIN(PB2), SCK_PIN(PB5), MOSI_PIN(PB3), MISO_PIN
// Arduino Pin       10           13            11             12
// MCP4921 DAC       SS           SCK           MOSI           n/a
#include <Spi.h>

// ATmega328P ADC
int analogPin = 0;     // analog input channel

// ADC analog input value
word sensorValue = 0;  // equivalent to unsigned int

// Byte of data to output to DAC
byte data = 0;

void setup() {
  //set pin(s) to input and output
  pinMode(analogPin, INPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(analogPin);   // comment out this line to test DAC
  // sensorValue = 0x0200;               // 0x03FF = Vref, 0x0200 = 1/2 Vref
  sensorValue = sensorValue << 2 ;       // 10 bit ADC to 12-bit DAC word
  // set SS pin low, beginning 16-bit (2 byte) data transfer to DAC
  digitalWrite(SS_PIN, LOW);
```

```
  // send high byte
  data = highByte(sensorValue);
  data = 0b00001111 & data;     // clear 4-bit command field (optional)
  data = 0b00110000 | data;     // 0=DACA, 0=buffered, 1=1x, 1=output enabled
  Spi.transfer(data);           // alt: shiftOut(MOSI, SCK, MSBFIRST, data);
  // send low byte
  data = lowByte(sensorValue);
  Spi.transfer(data);           // alt: shiftOut(MOSI, SCK, MSBFIRST, data);
  // set SS pin high, completing 16-bit transfer to DAC
  digitalWrite(SS_PIN, HIGH);
}
```

Bipolar Operation

source: Section 6.5 "Bipolar Operation," Microchip Technology Inc. MCP4921/MCP4922, 12-Bit DAC with SPI™ Interface, Microchip Technology document DS21897B

Bipolar operation is achievable using the MCP4921 by using an external operational amplifier (op amp). This configuration is desirable due to the wide variety and availability of op amps. This allows a general purpose DAC, with its cost and availability advantages, to meet almost any desired output voltage range, power and noise performance. Figure 5 illustrates a simple bipolar voltage source configuration. R1 and R2 allow the gain to be selected, while R3 and R4 shift the DAC's output to a selected offset. Note that R4 can be tied to VREF, instead of AVSS, if a higher offset is desired. Note that a pull-up to VREF could be used, instead of R4, if a higher offset is desired.



$$V_{OUT} = V_{REF} G \frac{D}{2^{12}}$$

$$V_{IN+} = \frac{V_{OUT} R_4}{R_3 + R_4}$$

$$V_O = V_{IN+}\left(1 + \frac{R_2}{R_1}\right) - V_{REF}\left(\frac{R_2}{R_1}\right)$$

G = Gain select (1x or 2x)
D = Digital value of DAC (0 – 4096)

Figure 4 Digitally-Controlled Bipolar Voltage Source.

An output step magnitude of 1 mV with an output range of ±2.05V is desired for a particular application.
1. Calculate the range: +2.05V – (-2.05V) = 4.1V.

2. Calculate the resolution needed: 4.1V/1 mV = 4100, Since 2^{12} = 4096, 12-bit resolution is desired.
3. The amplifier gain (R_2/R_1), multiplied by V_{REF}, must be equal to the desired minimum output to achieve bipolar operation. Since any gain can be realized by choosing resistor values (R_1+R_2), the V_{REF} source needs to be determined first. If a V_{REF} of 4.1V is used, solve for the gain by setting the DAC to 0, knowing that the output needs to be -2.05V. The equation can be simplified to:

$$\frac{-R_2}{R_1} = \frac{-2.05}{V_{REF}} = \frac{-2.05}{4.1} \qquad \frac{R_2}{R_1} = \frac{1}{2}$$

If R_1 = 20 kΩ and R_2 = 10 kΩ, the gain will be 0.5.

4. Next, solve for R_3 and R_4 by setting the DAC to 4096, knowing that the output needs to be +2.05V

$$\frac{R_4}{(R_3+R_4)} = \frac{2.05V+0.5V_{REF}}{1.5V_{REF}} = \frac{2}{3}$$

If R_4 = 20 kΩ, then R_3 = 10 kΩ