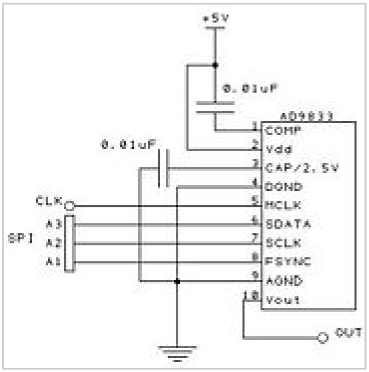
Waveform Generation with AD9833, and SPI

From Mech

Contents



Connection diagram for the AD9833 function generator.

Overview

The AD9833 is a programmable waveform generator capable of creating sine, triangular, or square wave outputs in a frequency range of 0 to 12.5 MHz. It has two 28-bit frequency registers and two 12-bit phase registers whose values can be used to calculate the frequency and phase of an output waveform. This chip is perfectly suited for use with the PIC 18F4520, as it uses SPI communication for its setup and control. A connection diagram for the AD9833 is given to the right. The MCLK pin is tied directly to the oscillator of the PIC, and the three SPI communication lines are connected to three I/O pins on the PIC (in this case pins A1, A2, and A3).

Setup

In order to establish communication between the PIC and the AD9833, we need to set up the SPI for the pins we wish to use. Here, we use the software support included with the CCS compiler (the PIC also offers hardware support, which we did not use. It should, however, be very similar). The following should be added to the header of your program.

```
#use spi(DO = PIN_A3, CLK = PIN_A2, ENABLE = PIN_A1, BITS = 16, MASTER, ENABLE_ACTIVE = 0, MSB_FIRST, IDLE = 1)
```

The first three arguments determine which three pins we will be using: DO is the "SDATA" pin of the AD9833, CLK is the "SCLK" pin, and ENABLE is the "FSYNC" pin. The next argument states that the max number of bits in one transfer is 16. The next argument specifies that the PIC is the master and the AD9833 is the slave. The AD9833's FSYNC pin is active low, and it accepts the most significant bit (MSB) of each transfer first. The SCLK pin is also specified to be kept high when not in use. After this initial setup, the SPI communication is quite straightforward. We simply use the function **spi_xfer()** whenever we want to send information via SPI to the AD9833.

AD9833 Control Register

In order to program the frequency generator, we need to write to its internal control register. This control register is used for resetting the chip, setting the mode of operation and selecting the frequency and phase registers on which the output is to be based.

Each of the 16 bits transferred has a meaning, and these meanings are described in the Table below. Note that the if a frequency register is being written to (first two bits = 01 or 10), the last 14 bits are the value of the write, and do not have the same meaning as specified in the table. Similarly, if a phase register is being written to (first two bits = 11), the last 12 bits are the value of the write. Only if the control register is being written to (first two bits = 00) are most of the meanings specified in the table applicable.

Bit	Significance
D15,D14(MSB)	10 = FREQ1 write, 01 = FREQ0 write, 11 = PHASE write, 00 = control write
D13	If D15,D14 = 00, 0 = individual LSB and MSB FREQ write, 1 = both LSB and MSB FREQ writes consecutively
	If D15,D14 = 11, 0 = PHASE0 write, 1 = PHASE1 write
D12	0 = writing LSB independently, 1 = writing MSB independently
D11	0 = output FREQ0, 1 = output FREQ1
D10	0 = output PHASE0, 1 = output PHASE1
D9	Reserved. Must be 0.
D8	0 = RESET disabled, 1 = RESET enabled
D7	0 = internal clock is enabled, 1 = internal clock is disabled
D6	0 = onboard DAC is active for sine and triangle wave output, 1 = put DAC to sleep for square wave output
D5	0 = output depends on D1, 1 = output is a square wave
D4	Reserved. Must be 0.
D3	0 = square wave of half frequency output, 1 = square wave output
D2	Reserved. Must be 0.
D1	If D5 = 1, D1 = 0. Otherwise 0 = sine output, 1 = triangle output
D0	Reserved. Must be 0.

Frequency and Phase Registers

In order for the chip to output at a frequency f_{out} and phase shift Φ_{shift} we need to have:

$$f_{out} = \frac{f_{MCLK}}{2^{28}} \times FREQREG$$

$$\Phi_{shift} = \frac{2\pi}{4096} \times PHASEREG$$

In other words, the output frequency and phase shift are not the values of the frequency or phase registers. They are related by the above equations, so the values you send to the registers need to be modified from the actual frequency and phase shift. For instance, if we sent 200 to the FREQ register and 100 to the PHASE register, and we were using a 20 MHz MCLK, we would output a 14.9 Hz wave with a phase shift of 0.15 radians.
The chip has two frequency registers (FREQ0 and FREQ1) and two phase registers (PHASE0 and PHASE1). By setting or resetting pins 11 (FSELECT) and 10 (PSELECT) of the control word we select which frequency register and which phase register to combine in order to produce the output waveform.

Writing to the Registers

The first two bits (most significant) of the SPI message select which of the registers is being written. The next bit is used in order to distinguish between the two phase registers, in case one of them is being written to.

Phase Registers

In order to write to a phase register, the most significant two bits of the serially transmitted word have to be 11. The next bit is the number of the phase register that is being written, following is a "don't care" bit and the rest represent the value to be written.

Control Register

Writing to the control register is done by setting the first two bits to 00. The other bits should be set according to the desired values found from the above table.

Frequency Registers

Writing to the frequency registers can be done in two ways: 1. Both the 14 MSB (most significant bits) and 14 LSB (least significant bits) of a frequency register will be altered - coarse tuning 2. Only the MSB or the LSB will be written - fine tuning In order to specify which kind of write is being performed we need to set bit 12(HLB) and 13(B28) of the control register accordingly. To specify which of the two frequency registers is being written, we have to set the first two bits preceding the data to either 01 for FREQ0 or 10 for FREQ1.

Example

In order to setup or update the AD9833 you first need to reset it, then transfer the frequency and phase information, then "unreset" it. For example, if we wanted to output a 20 kHz square wave with no phase shift, we would send the following:

```
spi_xfer(0b0010000101101000);
```

This resets the AD9833 (D8), tells it that we will be using the next two transfers to input both LSB and MSB (D13) of the FREQ0 register (D11), the internal clock is enabled (D7), the DAC has been put to sleep (D6), the output is to be a square wave (D5), and the square wave frequency is not to be divided by two (D3).

```
spi_xfer(0b0101100010010011);
spi_xfer(0b0100000000010000);
```

These transfers are writing to the value of the FREQ0 register (D15-D14), with the last 14 bits of each transfer representing first the LSB, then the MSB of the value. This writes a value of 0000000001000001100010010011 (268435 decimal) to the FREQ0 register. If we use the formula above, we see this corresponds to an output frequency of 20 kHz.

```
spi_xfer(0b1100000000000000);
```

This transfer is writing to the PHASE0 register (D15,D14,D13) and is writing a twelve bit value of zero (D11-D0).

```
spi_xfer(0b0000000001101000);
```

This transfer "unresets" the AD9833 (D8).

With a small variation in the values of the five transfers, all of the frequency and phase registers can be written to and output from the VOUT pin of the AD9833.

References

- AD9833 Datasheet, Analog Devices (http://www.analog.com/static/imported-files/Data_Sheets/AD9833.pdf)