MicroConverter® Technical Note – uC012
Expanding the number of DAC Outputs

Introduction

The Analog Devices family of MicroConverter products incorporate single and multiple voltage output DACs, as well as additional PWM outputs in some parts.

In certain applications additional analog output channels may be required. This problem is easily addressed by interfacing the MicroConverter to a discrete external multi-channel DAC. Interfacing the MicroConverter to Analog Devices' family of small footprint, SPI compatible, multi-channel DACs is an extremely easy way to expand the voltage output channels available in your system.

This technical note describes how to interface the ADuC814 to the AD5304/AD5314/AD5324 (hereafter referred as AD53x4), quad voltage output 8/10/12-bit DACs using a simple 3-wire SPI compatible interface.

AD53x4

The AD53x4 (Figure 1) is a quad 8/10/12-bit buffered voltage output DAC in a 10-lead micro SOIC package that operates from a single 2.5V to 5.5V supply consuming 500uA at 3V. Its on-chip output amplifiers allow rail-to-rail output swing to be achieved with a slew rate of 0.7V/us. A 3-wire serial interface is used which operates at clock rates up to 30 MHz and is compatible with standard SPI interface. For more details, please download the datasheet at
http://products.analog.com/products/info.asp?product=AD5304

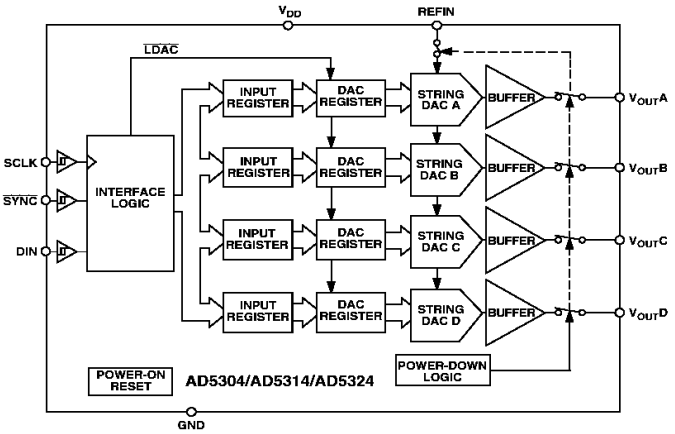

Figure1. AD53x4Functional Block Diagram

Hardware Interfacing

SPI Interface

Figure 2 shows a serial interface between the AD53x4 and the ADuC814 MicroConverter. Since the AD53x4 is a slave device thus the ADuC814 is configured as master to drive the SCLK of the AD53x4. The MOSI pin is configured as Master Output which drives the serial data line (DIN) of the DAC. The $\overline{\text{SYNC}}$ input of the AD53x4, effectively a chip select input, is derived from a bit-programmable pin on the port. In this case port line P3.4 is used.

When data is to be transmitted to the AD53x4, P3.4 is taken low. The AD53x4 input shit register is 16-bits wide and data is loaded into the device as a 16-bit word. Thus the AD53x4 requires 16-bits to be transferred per $\overline{\text{SYNC}}$ assertion. Since the ADuC814 transmits 8-bits per transfer, a second write operation is required to the AD53x4. After the bytes have been transmitted P3.4 is taken high to end the transmission. Data is transmitted MSB first. The DAC is updated at the completion of the write sequence.

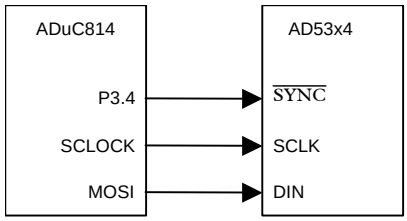

Figure 2. AD53x4 to ADuC814 Interface

Feeding the Reference Voltage from ADuC814

If desired, The ADuC814's internal reference can be fed to AD53x4. The ADuC814 provides an on-chip 2.5V precision bandgap reference. The internal 2.5V is factory calibrated to an absolute accuracy 2.5V ± 2.5% @ 25C. If this internal reference is used for AD53x4, it should be buffered at the CREF pin and a 100nF capacitor should be connected from this pin to AGND as shown in Figure 3. The typical noise performance for the internal reference, with 5V supplies is 150nV/Hz@1kHz and DC noise is 100mV p-p.

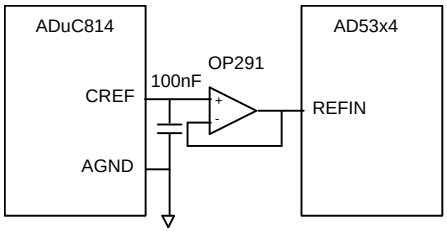

Figure 3. Using ADuC814 Reference Output

® MicroConverter is a Registered Trademark of Analog Devices Inc.
® SPI is a Registered Trademark of Motorola Inc.

Software Interfacing

List 1 is an ADuC814 code extract that is relevant to interfacing AD53x4.

The SPI interface is initialised in `init814` function, setting various mode parameters to meet AD53x4's SPI timing requirements.

The `ad53x4out` is the interface function that, first of all, formats the upper byte of 16-bit data from given parameters to be loaded to upper byte of AD53x4's input shift register (refer to AD5304/AD5314/AD5324 datasheet for the format details)

The byte is then sent using the `spiTX` function. Following this the lower byte is transmitted to complete 16-bit data transfer. Note that the `ad53x4cs` bit, which controls the $\overline{\text{SYNC}}$ of the AD53x4, is left asserted during the two consecutive byte transmission of the SPI.

The `spiTx` function which is called from `ad53x4out`, transmits the byte data contained in the `txDat` parameter by writing it to the SPIDAT register. The function then waits for the hardware to set the ISPI flag indicating that the transmission has been completed. Note that the ISPI bit is not cleared automatically by hardware so it must be cleared by software prior to writing to SPIDAT. Also note that the `spiTx` function must exit to `ad53x4out` after confirming completion of transmission to ensure that this function negates the $\overline{\text{SYNC}}$ after a data transfer.

```
sbit ad53x4cs = P3^4;                  // Chip select for ad53x4 (PORT3.4)

void init814(void)                     // Initialize internal peripherals
 {
  /* Initialize other peripherals here */

  /* Initialize SPI to talk to AD53x4 */
  CFG814 = 0x01;                       // Serial interface enable for P3.5..P3.7 pins
  SPICON = 0x38;                       // Enable SPI I/F as master, SCLOCK idle H,
                                       // advance MSB output, sclock=fcore/2=1.05Mhz
 }

void spiTx(unsigned char txDat)        // Transmit a byte data over the SPI
 {
  ISPI = 0;                            // Clear ISPI bit
  SPIDAT = txDat;
  while(!ISPI);                        // Wait until tx complete
 }

void ad53x4out(unsigned char adrs,     // A1,A0 bit of the contents
               bit pdN,                // PD bit of the contents
               bit ldacN,              // LDAC bit of the contents
               unsigned short dat)     // 12-bit output data
 {
  unsigned char txDat;

  ad53x4cs = 0;

  txDat = ((unsigned char) (dat>>8)) & 0x0f;
  txDat |= ldacN ? 0x10 : 0x00;
  txDat |= pdN ? 0x20 : 0x00;
  txDat |= (adrs<<6);
  spiTx(txDat);                        // Tx the upper byte

  txDat = (unsigned char) dat;
  spiTx(txDat);                        // Tx the lower byte

  ad53x4cs = 1;
}
```

List 1. AD53x4 relevant extract from companion code