ANALOG DEVICES

AN-759
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106 • Tel: 781/329-4700 • Fax: 781/326-8703 • www.analog.com

Expanding the Number of DAC Outputs on the ADuC8xx and ADuC702x Families

by Aude Richard

INTRODUCTION

The MicroConverter® family of products from Analog Devices incorporates both single and multiple voltage output DACs.

In certain applications additional analog output channels may be required. Therefore, the MicroConverter must be interfaced to a discrete external multichannel DAC. Interfacing the MicroConverter to Analog Devices' family of small footprint, SPI®-compatible, multichannel DACs is an extremely easy way to expand the voltage output channels available in your system.

This application note describes how to interface the ADuC814 or the ADuC7020 to the AD5304/AD5314/AD5324 (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 2.5 V to 5.5 V supply consuming 500 μA at 3 V. Its on-chip output amplifiers allow rail-to-rail output swing to be achieved with a slew rate of 0.7 V/μs. A 3-wire serial interface is used that operates at clock rates up to 30 MHz and is compatible with standard SPI interface standards. For more details, refer to the AD5304 data sheet.

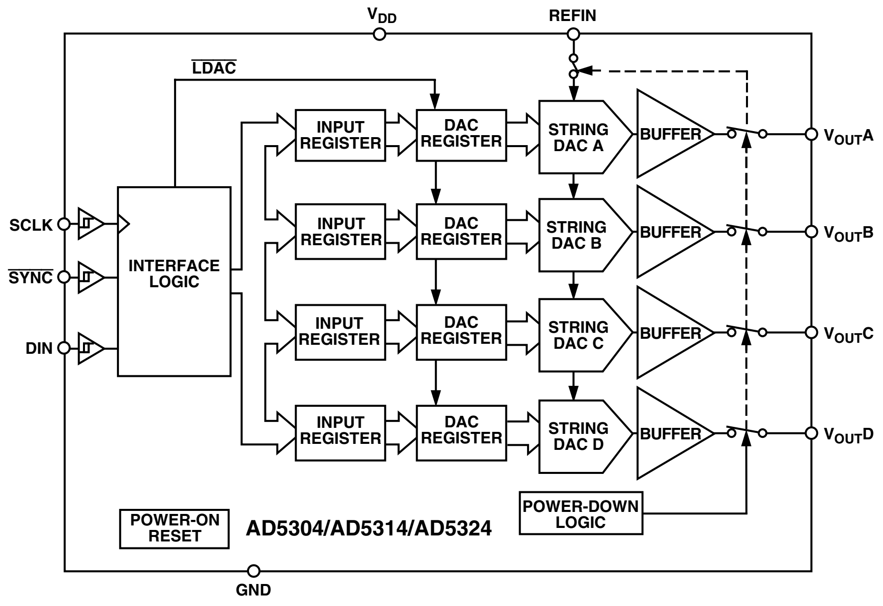

Figure 1. AD53x4 Functional Block Diagram

HARDWARE INTERFACING

SPI Interface

Figure 2 shows a serial interface between the AD53x4 and the MicroConverter. Since the AD53x4 is a slave device, the MicroConverter is configured as master, i.e., driving SCLK of the AD53x4. The MOSI (master out slave in) pin is connected to the serial data line (DIN) of the DAC. The SYNC input of the AD53x4, effectively a chip select input, is derived from a bit-programmable pin on the port of the ADuC814 (P3.4) or from a general-purpose I/O of the ADuC7020 (P1.7).

The AD53x4 input shift register is 16 bits wide and data is loaded into the device as a 16-bit word. It expects two bytes of data from the MicroConverter while SYNC is held low.

On the ADuC814, there is no slave select or chip select output and a GPIO must be used. On the ADuC7020, the SPI interface provides a slave select or chip select line that goes high after each byte transmitted, thus the need of a GPIO to control the SYNC line of the external DAC. Data is transmitted MSB first.

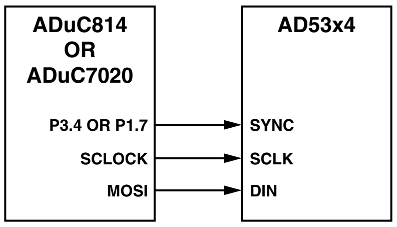

Figure 2. AD53x4 to ADuC814/ADuC7020 Interface

FEEDING REFERENCE VOLTAGE FROM THE MICROCONVERTER

If desired, the MicroConverter's internal reference can be fed to the AD53x4.

- The ADuC814 provides an on-chip 2.5 V precision band gap reference. The internal 2.5 V is factory calibrated to an absolute accuracy of 2.5 V ± 2.5%. If this internal reference is used for AD53x4, it should be buffered at the CREF pin to AGND, as shown in Figure 3. The typical noise performance for the internal reference with 5 V supplies is 150 nV/Hz @ 1 kHz and dc noise is 100 mV p-p.
- The ADuC7020 provides a 2.5 V on-chip reference, factory calibrated to an absolute accuracy of 2.5 V ± 10 mV. If this internal reference is used for the AD53x4, it should be buffered at the VREF pin and a 470 nF capacitor should be connected from this pin to AGND.

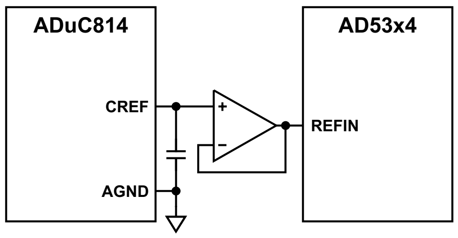

Figure 3. Using the ADuC814 Reference Output

SOFTWARE INTERFACING FOR ADuC814

List 1 is an extract that is relevant to interfacing the AD53x4 from companion code.

The SPI interface is initialized in init814 function, setting various mode parameters to meet the 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 the upper byte of the AD53x4's input shift register (refer to the AD5304/AD5314/AD5324 data sheet for 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, which is a SYNC of the AD53x4, is left-asserted during the two consecutive byte transmissions of the SPI.

The spiTx function, 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 ISPI flag to be set indicating that the transmission has been completed. Note that the ISPI bit is not reset automatically by hardware on transmission busy condition, so it must be set by software prior to writing to SPIDAT. Also note that the spiTx function must exit to caller after confirming completion of transmission to ensure that the caller negates the SYNC after data transfer, not in the middle of 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. Relevant Extract from Companion Code 53x4forADuC814.C

SOFTWARE INTERFACING FOR ADuC7020

List 2 is an extract of the companion code ADuC7020toAD53x4.c that is relevant to interfacing the AD53x4 to the ADuC7020.

The SPI interface is initialized in init702x function, setting various mode parameters to meet the 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 the upper byte of the AD53x4's input shift register (refer to the AD5304/AD5314/AD5324 data sheet for 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 P1.7, which is a SYNC of the AD53x4, is left-asserted during the two consecutive byte transmissions of the SPI.

The spiTx function, called from ad53x4out, transmits the byte data contained in the txDat parameter by writing it to the SPITX register. The function then waits until the SPIRX full bit (Bit 4) of SPISTA is set indicating that the transmission has been completed. Note that the SPIRX full bit must be cleared by software by reading a dummy data in SPIRX.

```
void init702x(void){
   GP1DAT = 0x80800000;            // Configure P1.7 as output
   GP1SET = 0x00800000;            // and pull /SYNC High
   GP1CON = 0x02020000;            // Configure P1.6 and P1.4 in SPI mode
   SPICON = 0x4B;                  // Configure SPI as Master, clock idles high
   SPIDIV = 0x14;                  // Set the DIV to run at 1.05Mhz
}

void spiTx(unsigned char txDat){
   SPITX = txDat;                  // Send txDat to the AD5304
   while(!(SPISTA & 0x08));        // Wait of end of transfer
   txDat = SPIRX;                  // Read in dummy data from SPIRX to prevent overflow
}

void ad53x4out(unsigned short address, unsigned short pdN, unsigned short ldacN, unsigned short dat){

   txDat1 = 0;                     // Reset txDat1 to 0
   txDat0 = 0;                     // Reset txDat0 to 0

   //configure the high byte to be sent
   txDat1 |= (address << 6);       // Store the address in the upper 2 bits of txDat1 (8 bit)
   txDat1 |= pdN ? 0x20 : 0x00;    // If pdN is set, copy a logic 1 into the next space,
                                   // if not copy a logic 0
   txDat1 |= ldacN ? 0x10 : 0x00;  // If ldacN is set, copy a 1 into the next space, if not copy a 0
   txDat1 |= (dat >> 4);           // Copy the first 4 bits of data into the last 4 bits of txDat1

   //configure the low byte to be sent
   txDat0 |= (dat << 4);           // Copy the last 4 bits of data into the first 4 bits of txDat0
   txDat0 &= 0xf0;                 // Insure the first 4 bits of data in txDat0 are empty

   //send the data
   GP1CLR = 0x800000;              // Pull /SYNC High
   spiTx(txDat1);                  // Send txDat1 to the SPI
   spiTx(txDat0);                  // Send txDat0 to the SPI
   GP1SET = 0x800000;              // Pull /SYNC Low
}
```

List 2. Relevant Extract from Companion Code ADuC7020to53x4.c