Interfacing digital audio

Patrick Gaydecki describes hardware and software issues relating to interfacing two new stereo audio converters to a microprocessor via serial links. Patrick's description revolves around a DSP56k processor, but the information will help anyone wanting to design with these high-performance, easy-to-interface audio d-to-a and a-to-d converters.

In this article is information for interfacing the CS5330/31 stereo analogue-to-digital converter and CS4330/31 stereo digital-to-analogue converter to a DSP56002 digital signal processor. Technical issues include the hardware interconnection, critical system timing signals and software modules for receiving and transmitting digital data.

The CS5330/31 is a stereo a-to-d converter while the CS4330/31 is a stereo d-to-a converter. Both are 18-bits wide. We chose the DSP56002 digital signal processor for our design example since it's an industry standard. There are many hardware and software strategies that could be used to interface these ICs, but we have narrowed our discussion down to the more straightforward techniques.

Analogue-to-digital conversion

The CS5330 and 31 are complete 18-bit stereo a-to-d converters. They perform anti-alias filtering, sampling, and analogue-to-digital conversion, generating binary data for both left and right inputs in serial form.

Alternate left and right channel data are transmitted via a single output. The sampling frequency can be adjusted infinitely between 2 and 50kHz, according to the frequency of a master-clocking signal.

These a-to-d converters use sigma-delta, shortened to ΣΔ, modulation with an oversampling rate equal to 128 times the equivalent sampling frequency. The sigma-delta stage is followed by digital filtering and decimation circuitry, which remove the need for an external anti-alias filter.

The linear-phase digital filter has a pass band to 21.7kHz, 0.05dB pass band ripple and greater than 80dB stop-band rejection. These devices contain a high-pass filter to remove DC offsets, which at a sampling rate of 48kHz, has a –3dB point of 3.7Hz.[3]

Digitising analogue signals

Complementary to the CS5330/31, the CS4330/31 are complete stereo digital-to-analogue converter systems. They include an interpolator, a 1-bit d-to-a converter and an output analogue filter.

Analogue signals generated by these devices are output to separate pins. In essence, these d-to-a converters perform the inverse operations to those of their a-to-d converter counterparts described above.

A digital interpolation filter first up-samples the incoming digital data by a factor of 128. A ΣΔ modulator then generates a 1-bit data stream, which is input to a linear analogue switched-capacitor low-pass filter. This enables infinite adjustment of the sampling frequency between 2 and 50kHz.

Output analogue signals require a simple first-order *RC* filter to eliminate images of the input signal at multiples of 128x the input sample rate.[4]

The CS5330 and the CS5331 differ only in the output serial data format. The CS4330 and the CS4331 differ only in the input serial data format. All devices are available as eight-pin plastic SOICs – a 5.28mm wide surface-mount package – with low power consumption, making them particularly attractive in power-conscious applications or in designs where space is at a premium.

Hardware interconnection

Figure 1 illustrates a simple interconnection strategy between the DSP56002, the CS533x and the CS433x. Both of the data converters are clocked by an external master-clocking signal, fed to their respective MCLK inputs. The serial data out pin of the CS533x, SDATA, connects to the SRD input of the processor.

In addition, the serial output connects to the ground rail by a 47kΩ resistor. This ensures that the CS5330/31 operates in master mode.

In master mode, the serial data clock, or bit clock, SCLK, and the left-right clock, LRCK, are generated as outputs

Patrick is with the Department of Instrumentation and Analytical Science at the University of Manchester Institute of Science and Technology (UMIST)

by the CS5330/31, derived from the the a-to-d converter clock MCLK input.

The LRCK output connects to the SC2 input of the DSP56002, used to accept the frame sync. The SCLK output of the CS5330/31 connects, via an inverter, to the SCK input of the DSP56002. The inverter is necessary because the DSP56002 samples the data present on the SRD input on the negative edge of the bit clock[5], whereas the data generated by the CS5330/31 are valid on the rising edge of the SCLK.

Since the CS5330/31 is operating in master mode, both the LRCK and the SCLK are fed directly as input signals to the CS4330/31, which is operating in slave mode. Because this uses the same clock protocol as the CS5330/31, no inversion of the SCLK signal is necessary.

Finally, the digital data generated as an output by the DSP56002 are fed from the STD pin to the SDATA input of the CS4330/31

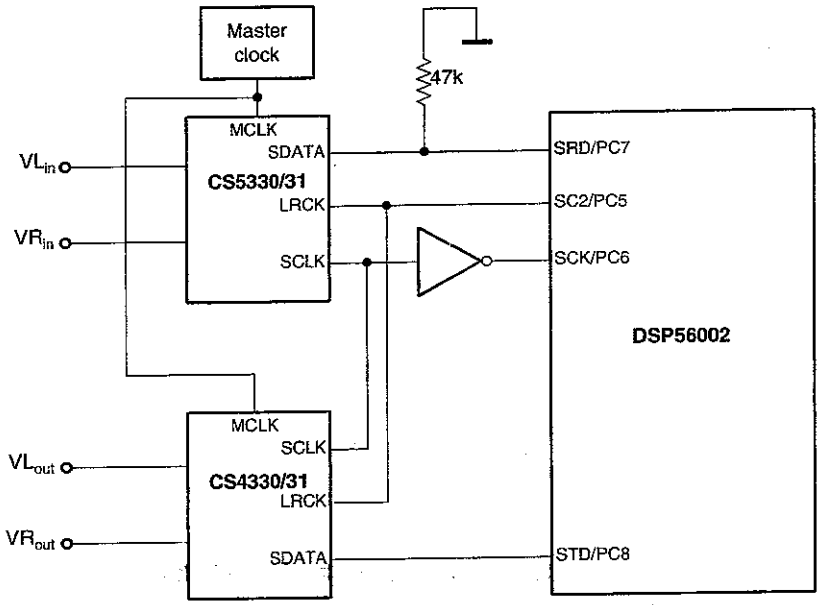

Fig. 1. Simple interconnection method between the DSP56002, the CS5330/31 and the CS4330/31.

Timing considerations and data formats

In master mode, the input clock rate of the CS5330/31 is 256 times the LRCK. This represents an over-sampling ratio of 128 for each channel.

In order to sample each channel at an effective rate of 48kHz for example, a clock frequency of 12.288MHz is required. Similarly, the CS4330/31 expects the same over-sampling ratio when driven by the CS5330/31, Fig. 1.

Figure 2 shows the output data format of the CS5330/31 as used here. Data for the left channel are output during the first half of the LRCK period, and data for the right channel are output during the second.

A total of 32 bits is generated during each half period, and the data are left justified, with the most-significant bit appearing first, on commencement of each half of the LRCK cycle.

Since the CS5330 has 18-bit resolution, the final 14 bits are redundant. This may be viewed another way; as far as the DSP56002 is concerned, the data for each channel occupy four time slots, each one eight bits in length. Hence eight time slots period characterise the entire LRCK.

The data format for the CS5331 is similar, except that it uses the I^{2}S format. Here, the first most significant bit is delayed by one bit clock period, for handshaking purposes. Figure 2 also shows the input data format expected by the CS4330 and CS4331.

In the case of the C4330, the data are right justified; hence the first 14 bits of any LRCK half-cycle are redundant. For the CS4331 with an externally supplied LRCK – the case discussed here – the data are left justified using I^{2}S format, i e the most-significant bit commences after a delay of one bit clock period.

Reading and writing in network mode

The SSI of the DSP56002 may be configured in normal mode or in network

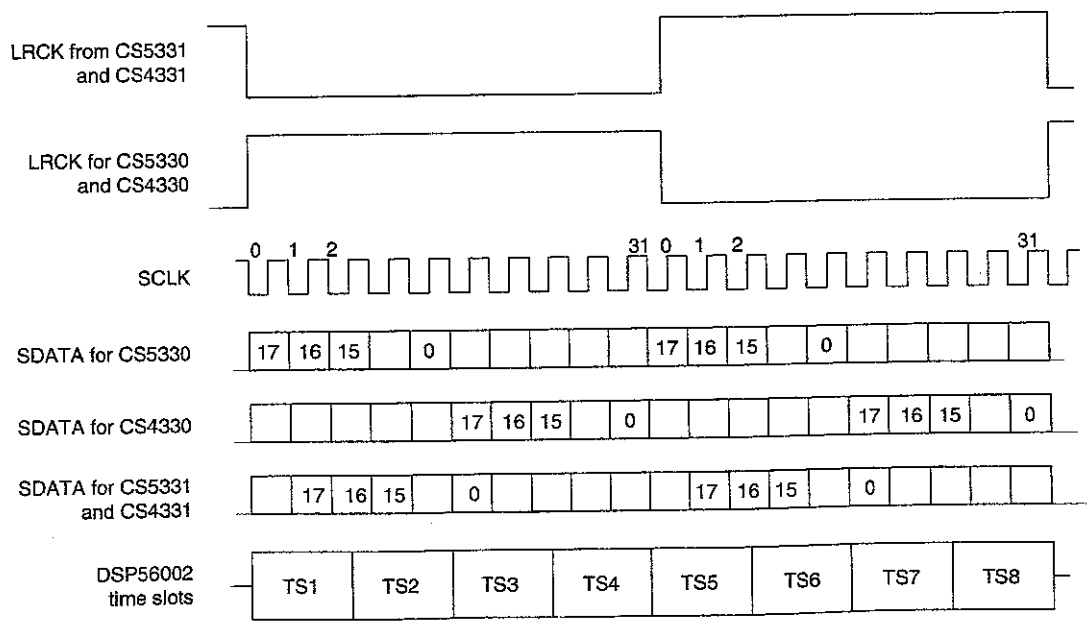

Fig. 2. Timing waveforms for the CS5330/31 and the CS4330/31. These assume the CS5330/31 is operating in master mode, and the CS4330/31 is clocked externally.

mode. In normal mode, a signal input to the SC2 pin is considered a framing signal, i.e. it frames the start and end points of one datum word.

Individual bits are clocked in using the SCK input as a bit synchronisation signal. In network mode, it is assumed that within any one frame, a number of data words are being transmitted in packets termed time slots (TS).

In the case outlined here, the SSI of the DSP56002 is configured in network mode, assuming a time-slot length of 8 bits. The SSI can be configured to accept words that are 8, 12, 16 or 24 bits in length.

In the CS5330/31, the data are left justified. Hence, since the device has 18-bit resolution, only the first three slots of any channel are required.

List 1. For network-mode polled communication, in which the DSP56002 operates as a slave, bits are set as follows.

Register	Bits	Function
CRA	0-7	Pre-scale modulus: set to 0 (only used if DSP is master).
CRA	8-12	DC: set to 7, since this equals the number of time slots minus 1.
CRA	13-14	WL: set to 0, since this gives a word length of 8 bits.
CRA	15	PSR: set to 0, as this is not needed in slave mode.

Hence CRA = $0000011100000000_2 = 700_{16}$

Register	Bits	Function
CRB	0-3	Set to 0, as they are not used.
CRB	4-5	Set to 0, external frame + bit clocks supplied by CS5330/31.
CRB	6	Set to 0, as MSB in/out is first and the LSB is last.
CRB	7-8	Set to 0, as the WL bit clock is used for both TX/RX.
CRB	9	Set to 1 for sync clock control, ignored in network mode.
CRB	10	Set to 0 for continuous clock.
CRB	11	Set to 1 for network mode.
CRB	12-13	Both set to enable RX and TX.
CRB	14-15	Not set since interrupts not required, i.e. polled-mode is used.

Hence CRB = $0011101000000000_2 = \$3A00_{16}$.

The DSP side

Motorola's DSP56002 is an advanced 24-bit fixed-point general-purpose digital signal processor, with 56-bit intermediate resolution. It has a highly parallel architecture, in that it treats program memory space separately from data memory space. This is known as Harvard Architecture.

Harvard Architecture has been extended further in the DSP56002 by sub-classifying the data space into X-data memory and Y-data memory – so-called Super Harvard Architecture. This is because many signal-processing algorithms use two distinct signal vectors. For example, FIR filters need memory to hold the incoming signal, and memory to hold the filter coefficients; FFT routines need memory to hold the real and imaginary Fourier components, and so on.

Each of the three memory areas – program, X and Y data – has its own data and address bus, and all of these connect to the outside world via bus multiplexers. The processor has 256 words of X-data RAM, each 24-bits wide, 256 words of Y-data RAM, and 512 words of program RAM (24 bits). This may not seem very much, but remember that the device is hardware oriented.

Operations that traditionally require many instructions can be implemented here using a single instruction, since the details are implemented in hardware. For example, a complete FFT routine needs only 40 words, i.e. 120 bytes.

In contrast, an FFT routine written on a conventional PC might require several thousand bytes. Since the 56002 uses hardware multipliers and pipelining, it can perform multiplication, accumulation and instruction fetching in a single instruction cycle, i.e. two clock cycles.

The fastest version of the 56002 can be clocked at 80MHz, which means it operates at 40 mega-instructions per second, i.e. 40MIPs. Also incorporated in the device are peripheral system components, including parallel and serial i/o ports. A host interface allows the processor to communicate with other computing devices and peripheral circuitry such as analogue-to-digital converters, digital-to-analogue converters, voice-band codecs and other devices.

For high-speed serial data transfer, its synchronous serial interface, or SSI, is often the preferred choice. It is readily compatible with many other products and it needs only a small number of signal connections and a minimum of glue-logic.[1,2]

Furthermore, only the first two bits of the CS5330, or first three bits of CS5331, time-slot 3, are used. These represent the two least-significant bits of the a-to-d converter word. The 8 bits from TS 1 and TS 2, together with the two bits from TS 3 are then combined into a single 18-bit word.

The procedure for writing to the CS4330/31 is similar, with the stages reversed. First, an 18-bit word is decomposed into three 8-bit time slots. For a CS4330, the most-significant bit commences at bit six of TS 2 for the left channel, TS 1 being blank. Lower-order bits follow accordingly in TS 3 and TS 4.

If instead, a CS4331 is being used, the most-significant bit commences at bit 1 of TS 1. The data are shifted one place right since bit 0 is ignored in I^2S format. Lower order bits follow accordingly in the remainder of TS 1, TS 2 and TS 3.

Polled-network mode

Whether in network or in normal mode, data may be clocked into the SSI using polled or interrupt-driven communication.

In a network-mode polled system, the program waits in a loop while the program tests the status of a flag in a status register. When true, this flag indicates the start of the LRCK sequence – the start of the left channel.

The program then proceeds to wait in another loop, testing the status of the flag that indicates a word has been received into the data register. When this is true, it reads and stores the data corresponding to that time slot – time-slot 1 initially.

The program now continues to wait for and read the data words corresponding to subsequent time slots. To use the SSI, it must first be enabled as a synchronous interface, since it has a dual function; it may also operate as a general-purpose i/o port.

Thus, the appropriate bits must be set in the port-C control register, PCC. In this case, the upper five bits must be set. Next, the device must be configured for a particular mode by loading the appropriate words into the SSI control registers A and B (CRA and CRB), located at $X:FFEC_{16}$ and $X:FFED_{16}$ respectively.

For network-mode polled communication, in which the DSP56002 operates as a slave, the bits are set as in **List 1**.

List 2 shows an assembly code fragment that reads data from the CS5330/31 and writes data to the CS4330/31 during all eight time slots using network-mode, polled communication.

Important flags are the receive-frame

sync flag RFS, bit 3 of the SSISR ($X:FFEE_{16}$), and the receive data register full flag RDF, which is bit 7 of the SSISR When RFS is set, it indicates the start of a frame or LRCK cycle When RDF is set, it indicates a datum word is present in the receive-data register, RX, located at $X:FFEF_{16}$.

The datum word may be transferred to an appropriate arithmetic register in the CPU or placed into memory. The code here simply reads the data repeatedly into register X0, but does nothing with it.

Once RX is read, the RDF flag is automatically reset. The transmit-data-register-empty flag TDE, bit 6 of the SSISR, is not required in this and all subsequent code described here. This is because the CS4330/31 is operating in slave mode, and data are transmitted by the DSP56002 within the time slots defined by the CS5330/31

Interrupt-driven network mode

Interrupt-driven i/o can be more efficient than polled i/o since the processor is only interrupted when a specific condition occurs; since the CPU is not locked in a polling loop, it can continue processing data until the status of a specific flag becomes true. When this occurs, the program automatically jumps to an interrupt service routine, which deals with the condition as required.

In order to use interrupt-driven i/o, bits 15 of the CRB, the receive-interrupt enable bit RIE must be set Register CRA remains the same. When the RIE bit is set, an interrupt is generated when the RDF flag becomes true, i.e. the receive-data register holds a new datum word. Hence,

$$CRB=1011101000000000_2=BA00_{16}$$

The necessary bits in the interrupt-priority register IPR, located at $X:FFFF_{16}$ must now be set for the SSI; these are bits 12 and 13, i.e the IPR must be loaded with 3000_{16}.

The interrupts must then be unmasked. This is achieved by ANDing the MR register with FC_{16}, i.e. the lowest two bits are set to 0, enabling all interrupts When an interrupt occurs, control passes to the interrupt vector whose address is at a specific area in program memory.

For the SSI receive data interrupt used here, the two-word vector space lies between $P:000C_{16}$-$P:000D_{16}$. Hence a JSR instruction is simply placed here to the start of an interrupt service routine whose purpose will be to read the data

In the program fragment shown in **List 3**, network-mode interrupt-driven communication is used to read the data as each time slot generates an interrupt However, the code as it stands is incomplete, since the program cannot distinguish the occurrence of the *first* time slot – i.e., it has no knowledge of the time slot order

In order to be of practical use, the code

List 2. Reading all eight time slots in network mode, polled configuration.

```
        ORG P:$0
        MOVEP #$1F0 X:$FFE1     ;Enable upper 5 bits of SSI
        MOVEP #$700,X:$FFEC     ;Configure CRA
        MOVEP #$3A00,X:$FFED    ;Configure CRB
LRCK    JCLR #3,X:$FFEE LRCK    ;Wait for LRCK sequence start
TS1     JCLR #7 X:$FFEE,TS1     ;Wait for first datum word  ie IS1

        MOVEP X:$FFEF,X0        ;Read 1st word into register X0 from a-to-d converter
        MOVE A1 X:$FFEF         ;Send data to d-to-a converter
TS2     JCLR #7,X:$FFEE,TS2     ;Wait for second datum word
        MOVEP X:$FFEF,X0
        MOVE A1 X:$FFEF
TS3     JCLR #7,X:$FFEE,TS3
        MOVEP X:$FFEF,X0
        MOVE A1 X:$FFEF
TS4     JCLR #7,X:$FFEE,TS4
        MOVEP X:$FFEF,X0
        MOVE A1 X:$FFEF
TS5     JCLR #7,X:$FFEE,IS5
        MOVEP X:$FFEF,X0
        MOVE A1 X:$FFEF
TS6     JCLR #7,X:$FFEE,IS6
        MOVEP X:$FFEF,X0
        MOVE A1 X:$FFEF
TS7     JCLR #7,X:$FFEE,IS7
        MOVEP X:$FFEF,X0
        MOVE A1 X:$FFEF
TS8     JCLR #7,X:$FFEE,IS8
        MOVEP X:$FFEF,X0
        MOVE A1,X:$FFEF
        JMP LRCK                ;All eight time slots read  Go and wait for
        END                     ;Start of next LRCK
```

List 3. Reading/writing of all eight time slots in network mode, interrupt-driven configuration.

```
        ORG P:$0
        JMP MAIN                ;Skip interrupt vector space
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        JSR ISR                 ;Jump to ISR  Executed when RDF=1
MAIN    MOVEP #$1F0 X:$FFE1     ;SSI mode
        MOVEP #$700,X:$FFEC
        MOVEP #$BA00 X:$FFED    ;Enable REI for SSI
        MOVEP #$3000 X:$FFFF    ;IPR for SSI
        ANDI #$FC,MR            ;Unmask interrupts
LOOP    JMP LOOP                ;This loop represents the main program
ISR     MOVEP X:$FFEF,X0        ;Start of interrupt routine.  Read data
        MOVE A1 X:$FFEF         ;Write to d-to-a converter
        RTI                     ;Return from interrupt
        END
```

List 4. Reading/writing of all eight time slots using interrupts and LRCK identification.

```
        ORG P:$0
        JMP MAIN                ;Skip interrupt vector space
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        JSR ISR                 ;Jump to ISR  Executed when RDF=1
MAIN    MOVEP #$1F0 X:$FFE1     ;SSI mode
        MOVEP #$700,X:$FFEC     ;
        MOVEP #$BA00 X:$FFED    ;Enable REI for SSI
        MOVEP #$3000 X:$FFFF    ;IPR for SSI
        ANDI #$FC MR            ;Unmask interrupts
LOOP    JMP LOOP                ;This loop represents the main program
ISR     MOVEP X:$FFEF,X0        ;Start of interrupt routine.  Read data
        INC A                   ;Increment the IS counter
        JCLR #3 X:$FFEE IGNORE  ;Find LRCK start  ie RFS=1
        CLR A                   ;If frame start, reset TS counter (a)
IGNORE  MOVE A1 X:$FFEF         ;Write to d-to-a converter
RTI                             ;Don't reset counter if not start of LRCK
        END
```

List 5. For single channel, normal-mode polled communication, in which the DSP56002 operates as a slave, bits are set as follows:

Register	Bits	Function
CRA	0-7	Pre-scale modulus: set to 0 (only used if DSP is master)
CRA	8-12	DC: set to 0, since this equals number-of-words/frame–1.
CRA	13-14	WL: both set to 1, giving a word length of 24-bits
CRA	15	PSR: set to 0, as this is not needed in slave mode.

Hence CRA=0110000000000000$_2$=6000$_{16}$.

Register	Bits	Function
CRB	0-3	Set to 0 as they are not used.
CRB	4-5	Set to 0; ext. frame and bit clocks supplied by CS5330.
CRB	6	Set to 0; MSB i/o is first and the LSB is last.
CRB	7-8	Set to 0, as the WL bit clock is used for both TX/RX
CRB	9	Set to 1 for synchronous clock control.
CRB	10	Set to 0 for continuous clock.
CRB	11	Set to 0 for normal mode.
CRB	12-13	Both set to 1 to enable RX and TX.
CRB	14-15	Not set since interrupts not required; polled-mode is used.

Hence CRB=0011001000000000$_2$=3200$_{16}$.

List 6. Reading a single channel of the CS5330 and writing to the CS4331 in normal mode.

```
        ORG P:$0
        MOVEP #$1F0,X:$FFE1     ;SSI mode
        MOVEP #$6000,X:$FFEC    ;Configure CRA and CRB for 24-bit
        MOVEP #$3200,X:$FFED    ;data  normal mode
LOOP1   JCLR #7,X:$FFEE,LOOP1   ;Wait for 24-bit word
        CLR A
        MOVE X:$FFEF,A1         ;Read data into a
        LSR A                   ;Shift right 1 place for CS4331
        MOVE A1,X:$FFEF         ;Send to CS4331
JMP LOOP1
END
```

shown in **List 3** should be capable of determining the start of the LRCK. This may be achieved by testing the status of the RFS flag when an interrupt occurs. If it is set, this means the LRCK has just commenced a new cycle and the data correspond to the first time slot.

A time-slot counter that records the current time slot should then be initialised to zero. If RFS is not set, then data corresponding to a later time slot are present in the receive data register. In this case, the time-slot counter should be incremented.

This scheme is very similar to the one described on page 6-137 of the DSP56002 Digital Signal Processor User's Manual[1]. **List 4** provides a code fragment which uses interrupts and polling of the RFS to both determine the start of the LRK cycle and read the data with a positional knowledge of the time slots.

Reading and writing a channel in normal mode

The above examples omitted code details dealing with recombination of data after its reception from the CS5330/31, processing and decomposition of the data prior to transmission to the CS4330/31.

Recombination involves weighting and summing the 8-bit words associated with, in this case, the three relevant time slots. Decomposition involves shifting to isolate the appropriate segments of the 18-bit word, and sending these segments out as eight-bit words corresponding once more to the time slots appropriate for the CS4330/31. This requires processing time and may limit the speed of the DSP environment – especially in situations where critical real-time operations are being performed.

One way of removing this difficulty is to treat the CS5330 as a single-channel device – i.e. left channel only – using it in normal mode. Similarly, data can be sent to the right-channel only of a CS4331. The channels switch because the LRCK signal is inverted for the CS4331, Fig. 2.

Since the only difference between the CS5330 in master mode and the CS4331 when clocked externally is a single bit-pulse delay – due to the I^{2}S format involved – a single right-shift operation is all that is required to make the input datum word compatible with the output.

When operating in normal mode, the CRA and CRB registers must be configured to accept 24-bit data. Trailing bits are simply ignored by the system. The LRCK signal is treated as a true framing signal, with only the leading edge of significance to the SSI.

Once this framing signal has been detected, the SSI clocks in all 24 bits of the datum word generated by the CS5330 which may then be processed, shifted one place right and transmitted to the CS4331. For normal mode polled communication, in which the DSP56002 operates as a slave, the CRA and CRB are set as in **List 5**.

The code fragment in **List 6** allows the DSP56002 to read in a full 18-bit word from the CS5330, shift it one place right, and output it to the CS4331. If processing is to be performed, the code would be included immediately after reading the data from the input register.

In summary

The interfacing strategies described here represent a small number of variations within a wide range of possibilities. In particular, it is possible to configure the DSP56002 as the master device.

Alternatively the expected data input format of the CS4330/31 may be altered by changing its default configurations through the use of a control word. This word is sent via the SCLK input. This is described in detail in reference 4.

However, if you do not wish to change these, the code given here should provide a starting point for successfully exchanging data between these devices. ■

References

1. *DSP56002 Digital Signal Processor User's Manual* Motorola Inc. Document DSP56002UM/AD (1993)
2. *DSP56000 Digital Signal Processor Family Manual* Motorola Inc. Document DSP56KFAMUM/AD (1995)
3. *CS5330/31 8-Pin Stereo A/D Converter for Digital Audio* Cirrus Logic Inc. Data sheet, document DS138F1 (1997)
4. *CS4330/31/33 8-Pin, Stereo D/A Converter for Digital Audio* Cirrus Logic Inc. Data sheet, document DS136F1 (1997)
5. *DSP56002/DSP56002 24-bit Digital Signal Processor*, Motorola Inc. Data sheet document DSP56002/D (1995)